Institute of Architecture of Application Systems

Integrating Compliance into Business Processes: Process Fragments as Reusable Compliance Controls

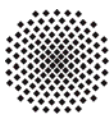
David Schumm, Frank Leymann, Zhilei Ma, Thorsten Scheibler, Steve Strauch

Institute of Architecture of Application Systems,
University of Stuttgart, Germany
{schumm, leymann, ma, scheibler, strauch}@iaas.uni-stuttgart.de

BibTeX:

```
@inproceedings{SchummLMS10,
  author    = {David Schumm and Frank Leymann and Zhilei Ma and
               Thorsten Scheibler and Steve Strauch},
  title     = {Integrating Compliance into Business Processes:
               Process Fragments as Reusable Compliance Controls},
  booktitle = {Proceedings of the Multikonferenz Wirtschaftsinformatik,
               MKWI'10, 23-25 February 2010, Göttingen, Germany},
  year      = {2010},
  publisher = {Universitätsverlag Göttingen}
}
```


The original publication is available at
http://webdoc.sub.gwdg.de/univerlag/2010/mkwi/



Universität Stuttgart
Germany

Integrating Compliance into Business Processes

Process Fragments as Reusable Compliance Controls

David Schumm, Frank Leymann, Zhilei Ma, Thorsten Scheibler, Steve Strauch

Institute of Architecture of Application Systems,
University of Stuttgart

1 Introduction

Companies increasingly have to pay attention to compliance concerns addressing business processes. Flexibly reacting to changing requirements coming from laws, regulations, and internal guidelines becomes a necessary part of business process management. Compliance refers therein to the entirety of all measures that need to be taken in order to adhere to laws, regulations and guidelines within the company, subsumed as compliance sources (Daniel et al., 2009, p. 1). In order to comply, companies need to perform profound changes in their organizational structure, business processes, IT systems and applications that drive their business. At this, process-awareness is basic prerequisite for ascertaining whether existing business processes are set up to operate in a compliant manner (Caprasse et al., 2008, p. 14). Among other steps, a compliance office can be installed, role-management is being established, and controls are integrated into particular processes. Consequently, compliance also has an impact on IT systems, applications and supporting infrastructure, as they have to support the execution, monitoring and checking of compliance issues.

Yet, in the field of Business Process Management (BPM) there is currently no agreed upon solution for enabling a flexible management of compliance requirements resulting from the interpretation of various compliance sources. There is no ultimate solution that allows to integrate compliance into processes or IT systems, and which specifies, how the execution of processes can be monitored to validate a compliant execution. The integration of compliance thus often results in hard-wired changes and tangled code. Another shortcoming of current solutions is that they do not address the whole compliance management life cycle (Daniel et al.,

2009, p. 3). Our approach overcomes the drawbacks of the current solutions and aims at ensuring a faster and more consistent specification and integration of compliance controls in business processes. A compliance control in this sense is the mechanism that actually realizes a compliance requirement. In this paper we exemplify realizing compliance requirements employing the notion of process fragments, and we show its characteristics and its practical application using a scenario common in industry. We discuss how a fragment can be identified, which design considerations need to be taken into account, we discuss efficient storage and retrieval, and which ways of integration into business processes are feasible.

A process fragment in our work is defined as a connected graph, however with significantly relaxed completeness and consistency criteria compared to an executable process graph. A process fragment is made up of activities, activity placeholders (so-called regions) and control edges that define control dependency among them. A process fragment may define a context (e.g., variables) and it may contain a process start or process end node, but it is not required to do so. It may contain multiple entering and leaving control edges for integration into a process or with other process fragments. A process fragment has to consist of at least one activity and there must be way to complete it to an executable process graph.

Therefore, a process fragment is not necessarily directly executable and it may be partially undefined. Additionally, for usage as reliable compliance control, a process fragment (i) may be parameterizable in order to mark points of variability and (ii) its contained placeholders may be constrainable. This allows explicitly declaring which parts of a process fragment may be changed and how the fragment may be adjusted and composed, while still realizing the compliance requirement that it has been designed for. (iii) Furthermore optional and mandatory entries and exits of a process fragment should be distinguishable, in order to declare how the fragment has to be wired within a process for ensuring compliant behaviour.

However, process fragments are not capable of realizing all compliance requirements concerning business processes. Process fragments are capable of specifying or constraining control flow and data flow within a process, but to the best of our knowledge they cannot ensure compliance of the applications and humans which are involved in the process. For instance, a compliance requirement that demands encrypted storage of customer billing data for at least seven years cannot be realized by a process fragment, as this requirement is related to a database that is external to the process. For specifying compliance controls that are not expressible as process fragments we use textual annotation, which we see as additional type of compliance controls.

This paper is structured as follows: we begin with related work on compliance and reusability in the area of business processes in Section 2. Motivated by an example shown in Section 3, we describe in Section 4 the approach for tackling compliance with the aid of process fragments. Finally, Section 5 summarizes the paper and characterizes future work.

2 Related Work

First of all we take a look at the related work in the field of augmenting business processes with compliance. Ly et al. (2008) analyze the requirements for support of semantic constraints in business process management systems and give a survey on existing approaches. The main outcome of the survey is, that existing approaches either consider the validation at process template level at design time or compliance monitoring at process instance level at runtime. The recommendation is to ensure compliance over the complete BPM life cycle, called life cycle compliance. The approach and concept for augmenting a business process with compliance that we introduce in this paper supports guarantee of compliance during all life cycle phases (cf. Section 4).

Several approaches exist, that cover only one specific compliance domain. For instance, Küster et al. (2007) consider compliance of business process models in respect of data object life cycles. An object life cycle is defined as a model that captures allowed states and state transitions for a particular data object. Process models, which comply with object life cycles employed for generation, are generated using automata theory. Further approaches address a typical compliance requirement, which is role-management and with it support for segregation of duty, such as described by Wolter and Schaad (2007). The approach of Sadiq et al. (2007) aims at augmenting a business processes with annotations concerning compliance, called control objectives in this work. The paper provides a research agenda and proposes the use of Formal Contract Language (FCL) as formalism to express normative specifications. The control objectives are stored separated from the process in a control objectives repository.

As a recent analysis by the European Project COMPAS (2009) has shown, several approaches for improving reusability have been proposed so far which can be applied to BPM, for instance the concepts of sub processes, worklets, reference models and business rules (cf. European Project COMPAS, 2009, pp. 46). The analysis showed that each of the analyzed approaches has certain advantages, but also drawbacks in its application in practice. Many approaches for reuse such as reference models are quite heavyweight, others such as sub processes are rather rigid and monolithic. Eberle et al. (2009) propose process fragments as a lightweight approach for reusability. They employ process fragments for the representation of small pieces of process knowledge and discuss an algebraic foundation for the composition of fragments into more complex structures. However, the focus of almost all these approaches has been set on reuse and none of them addresses in particular compliance requirements. Therefore we need an approach that combines the advantages of the available work, and modifies and extends the given concepts in order to define a solution for reusability *and* compliance in BPM.

3 Motivating Scenario

In the following we present a case study that motivates our approach. Due to a changed regulation a bank is forced to comply with a new compliance requirement in its loan approval process. For credits that exceed one million Euros the branch manager needs to double check whether the credit may be granted or whether the risk is too high. The officer responsible for realization of the compliance requirement instructs the process designer who shall adapt the process model according to the new requirement. The process designer changes the existing process model accordingly and adds necessary activities.

As the integration of an additional check is a frequent requirement it is feasible for reuse. In order to enable later reuse the process designer extracts the added activities, deletes context specific information that is only related to this particular process, abstracts from details, defines parameters and thereby creates a new process fragment. The designer annotates the fragment with metadata that enables efficient retrieval. The metadata also describes the purpose of the fragment, and may give examples for its usage. The fragment is then stored in a repository and ready for reuse in a different context. Figure 1 illustrates the fragment which the process designer has created.

Sometime later, another compliance requirement needs to be addressed, that is originated in the bank's internal quality policy: During the marketing material creation process an additional step for quality assurance shall be added. The process designer queries the process fragment repository in order to find a fragment that does the job. The previously created fragment is found, the only thing the process designer needs to do now is to tailor the process fragment to the needs of the given context, which includes setting parameters for the check, defining if the check should occur in any case, and so on. Subsequently the tailored process fragment is being integrated into the process model.

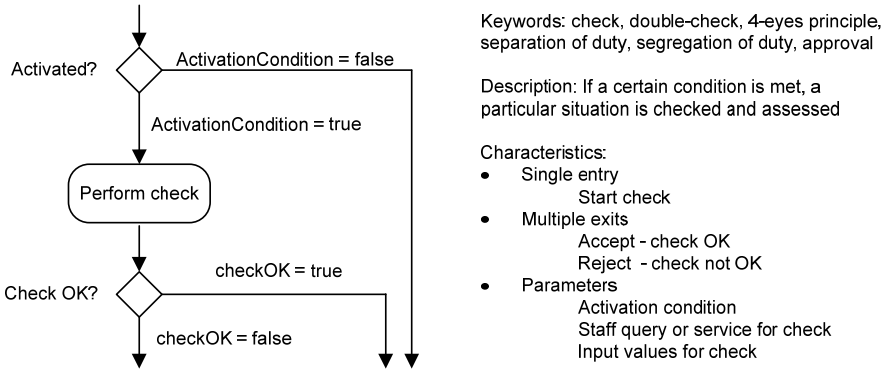


Figure 1: Designed Process Fragment for Checking

4 Compliance Fragment Life Cycle

Identification, design, and application of process fragments for compliance are not trivial. Besides the comprehensive understanding of the business domains, users need a well-defined life cycle for guidance. The life cycle shown in Figure 2 consists of four main phases: identification, design, storage & retrieval, and application. The application phase is further divided into three alternative scenarios of applying process fragments for compliance in design and run-time. In the following we discuss each phase of the life cycle in detail.

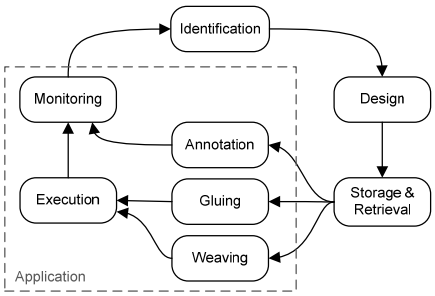


Figure 2: Compliance Fragment Life Cycle

4.1 Process Fragment Identification

In Section 1 we have introduced a process fragment as a connected and typically incomplete process graph with significantly relaxed consistency and completeness criteria. Furthermore, a process fragment represents a meaningful part of a process that has the potential to be reusable in various different process models. For instance, the implementation of a specific compliance requirement might be such a part, which is essential for the usage in the field of compliance.

Currently there exists no ideal way for finding appropriate fragments. The identification can for example, be achieved using mining techniques to find often repeating execution patterns in already executed process instances. Or, a domain expert analyzes already implemented process models 'by hand' and recognizes repeating patterns in the processes which indicate the potential of reusability. Both approaches are so-called top-down approaches. The other way round is building fragments from scratch (bottom-up). Nevertheless, no automatic tooling exists for identifying candidates that might be *meaningful* fragments. In fact, this identification is based on best-practice: experience in modelling and the knowledge of which requirements occur more than once.

4.2 Process Fragment Design

As mentioned in the previous subsection, we distinguish two approaches when dealing with the design of process fragments, bottom-up and top-down. In the bottom-up approach a fragment is developed from scratch, based on the specification of its purpose and the functional and non-functional requirements it shall meet. For the latter, top-down approach a process fragment is being extracted from an existing process model in which it is implemented. First of all the extraction of the activities that are identified as relevant for a specific purpose, their control structures and their context (such as variables or partner links) takes place. The sub-graph obtained thereby (i.e., the process fragment) needs to be customized afterwards. Activities may need to be reordered, attributes on activities may need to be omitted in order to hide confidential or process-specific information. Parameterization techniques and also the extraction of business rules can be applied in this phase to further increase of the reusability of the fragment.

4.3 Process Fragment Storage and Retrieval

Process fragments created in the design phase should be stored in a consistent and systematic manner, so that they can be retrieved later for reuse. A repository (Bernstein and Dayal, 1994) provides feasible capabilities for this task, which are well applicable to process fragments. The repository assigns a Universal Unique Identifier (UUID) (e.g., as proposed by Leach, 2005) to each process fragment being stored. The usage of UUID allows to annotate process fragments to a given process model by adding a reference from the UUID of the process model to the UUIDs of the process fragments respectively. In addition, it allows users to retrieve a process fragment by simply passing the corresponding identifier to the repository.

The usage of unique identifiers also allows linking of the process fragments that realize a particular compliance requirement to their source, e.g., to a particular section in an electronic version of a legal document. Maintaining such links is essentially important in order to flexibly react on changes in laws. An example query request could be: Retrieve all the process fragments that have been annotated to business processes and encode a part of Basel II (Basel Committee on Banking Supervision, 2006). Besides using identifiers to retrieve process fragments, the metadata associated with the stored process fragments as well as the structures of process fragments can be used as query criteria. The structure of a process fragment refers to the way in which process activities are connected together, arranged or organized (cf. Ma and Leymann, 2009, pp. 3).

4.4 Application of Process Fragments in the Field of Compliance

Preceding the application of process fragments, a compliance assessment is performed. This assessment is a non-technical step when dealing with compliance: Compliance experts, i.e. lawyers and consultants assess requirements from various compliance sources. Thereby they analyze and interpret which requirements are relevant for a particular process in a company. The effective guidelines resulting from this step, i.e. the concrete compliance requirements can be distinguished into two distinct groups.

The first group of requirements specifies *what* should be done. Either this requires the integration of additional activities or changes to a given process model. In the following we discuss two different ways for the integration of compliance related activities: process fragment gluing and process fragment weaving. The second group of requirements defines *how* something should be done. Such requirements necessitate the employment of annotations. Annotations are typically used to state how certain things should be executed. But they can also be used to state, what 'must occur', what 'must not occur', 'what may occur' etc. in the execution of a process.

4.5 Process Fragment Gluing

When referring to gluing, we have the most straightforward approach for the integration of compliance in mind. It means that the process fragments that realize certain requirements are physically copied into the original process model (cf. Figure 3a). For traceability and also for maintenance reasons a link between the process fragments (cf. Figure 3b) and the augmented process model (cf. Figure 3c) can be established to distinguish between the original parts of the process and the compliance augmentation. Furthermore a linkage to the compliance source can thereby be enabled.

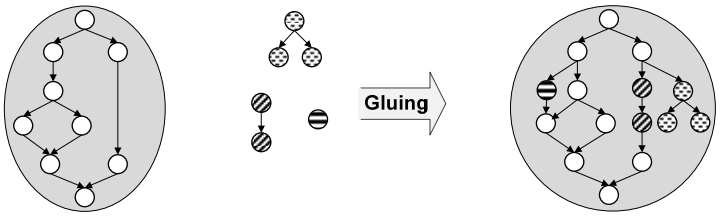


Figure 3: Process Fragments Gluing

This approach can be considered as best concerning execution performance, but it comes with several shortcomings. One issue when gluing the compliance fragments into a process is process pollution. Especially in relation to compliance, certain requirements (e.g., logging) might pollute a process model with activities

which are not relevant for the actual work that is performed in the process. The notion of views on processes such as described in Bobrik et al. (2006) has brought up a concept that allows decreasing the negative effect of such pollution, at least at design time. A view allows abstracting from undesired details, applied to our case it allows fading out glued-in compliance fragments and thereby showing the original process model.

When gluing is applied for integrating compliance into processes, all process models that are affected by a change have to be adapted accordingly and be redeployed, which means significant effort. Another issue is that the process fragments can be modified after they have been glued into the process and thus not realizing the compliance requirement anymore that they have been designed for. These shortcomings make a more loosely coupled approach preferable, thus we discuss weaving as a potential alternative. However some compliance requirements do not only necessitate additional activities, but they also require physically redesigning a process model (e.g., when the ordering of activities has to be changed), so gluing is not avoidable in any case.

4.6 Process Fragments Weaving

Aspect-oriented Programming (AOP) was introduced as a programming technique by Kiczales et al. (1997) which addresses the problem of integrating cross-cutting concerns e.g., logging, into applications. Karastoyanova and Leymann (2009) have shown the applicability of aspect-orientation in business processes. We argue that by employment of aspect-orientation, business processes can be customized to meet compliance requirements. Additional fragments such as auditing activities can be weaved into a process. This means, the original process model (cf. Figure 4a) stays untouched, however its execution is modified. An 'aspect weaver' handles the process instances and the aspects (i.e. the compliance fragments in Figure 4b) and composes them to a process augmented with compliance controls (cf. Figure 4c).

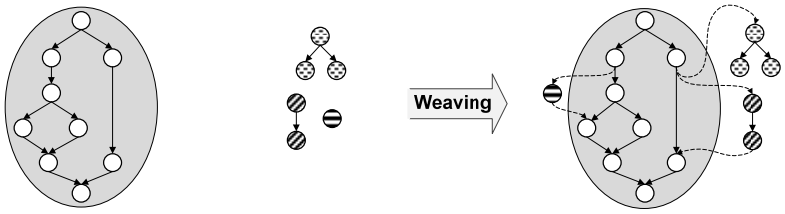


Figure 4: Process Fragments Weaving

Depending on the specific implementation, the aspect weaving can be performed at compile time (static) or run-time (dynamic), the latter one being more powerful as single instances can be treated differently, but also more complex. The clear separation of concerns using AOP, i.e. the separation of the actual work performed in a process and the compliance controls, comes along with increased

technical difficulty, e.g., debugging is becoming way more complex. This technique demands for well-defined programming guidelines and it also requires further elaboration of the technical details. Our approach is to shape the compliance controls as process fragments and thereby achieve overall compliance in a loosely coupled and flexible manner without polluting the original process model.

4.7 Process Fragments Annotation and Monitoring

The common approach to annotate compliance information to a process model is using textual annotation on either a high level of abstraction (cf. Figure 5), on a more technical level, or even on a formal level as proposed by Sackmann and Kähmer (2008). Compliance requirements can basically be stated in any language that is convenient for the domain they are stemming from. When employing domain-specific languages as discussed by Oberortner et al. (2009) it is even conceivable to have various different languages used in the annotation, for instance one for security considerations and another for data storage policies.

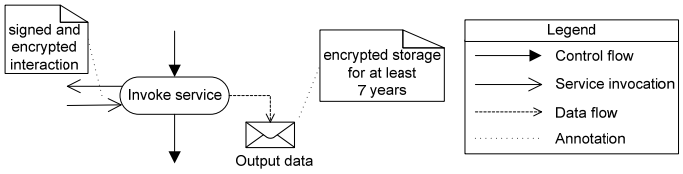


Figure 5: Usage of Textual Annotations

This approach is feasible for many compliance requirements, but we believe it is inadequate for stating complex compliant behaviour in terms of control flow or data flow. For this reason we propose the annotation by process fragments (cf. Figure 6a) in addition to the textual annotation (cf. Figure 6c) as we have described in former work (Daniel et al., 2009, p. 6). This allows describing the required behaviour also in terms of control flow or data flow of a process model in a way that is practicable for a process designer and that is also feasible for further machine-processing, especially for process monitoring.

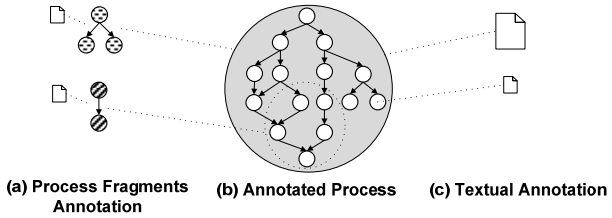


Figure 6: Usage of Process Fragments as Constraint Language

Note that for this application the activities related to a certain requirement (that is realized by a process fragment) do not have to be contained in the process model (cf. Figure 6b) at all. Assume the case that a process fragment expresses that two particular activities are executed. This fragment can be annotated to a process

with the additional information (i.e. the annotation of the fragment itself), that this behaviour must not occur. In case the behaviour actually occurs, it will be detected in monitoring, but it is also possible that this behaviour never occurs as the process is already modelled to comply with this requirement. Thereby either compliance is achieved by design, or incompliance is detected and reported by monitoring or mining which leads to redesign and consequential to compliance of future executions of the process.

Employing process fragments during monitoring (cf. Figure 7) requires a model transformation of the process fragments that are annotated to the process, and hereby also taking their annotation into account, into statements that can be checked during monitoring, for example into complex event processing rules. Therefore any information that is required in order to properly assess compliance of the execution has to be contained in the process fragment or its annotation. The concrete mapping of process fragments to statements which can be checked during monitoring is ongoing collaborative work in the COMPAS project.

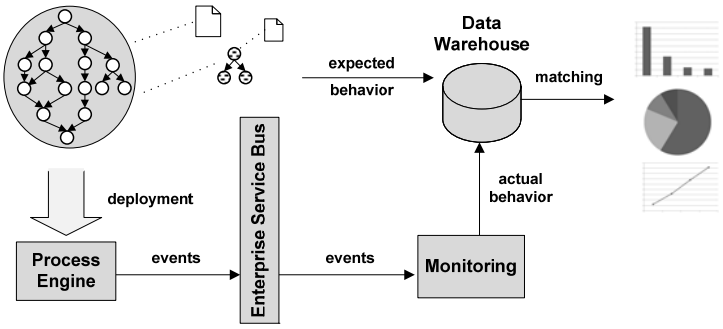


Figure 7: Usage of Process Fragments in Monitoring

5 Conclusion and Future Work

Our approach has three significant contributions: Firstly, the approach is providing support for (re-)integrating compliance requirements into a process. We achieve this by presenting a way to integrate compliance requirements that are realized by process fragments. Secondly, we envision process fragments as novel language for stating constraints concerning the execution behaviour of process instances. This concept allows precisely stating compliant behaviour in terms of control flow and data flow. Thirdly, by employing transformations this constraint language can be used in monitoring in order to achieve a correlation of the actual execution and the expected, i.e. compliant execution, based on events which actually occur. The concept presented in this paper allows linkage of business processes, process fragments, textual annotations, execution information and compliance sources with each other. This linkage supports demonstrating a lawful implementation of compliance requirements in case of an audit.

When using process fragments for compliance in business process modelling, process fragments have to be stitched together, e.g., in case of gluing. A (semi-) automatic tooling should help users to accomplish such tasks easier. Our next step is the creation of a comprehensive metamodel for process fragments with enriched metadata on the process fragments, such as pre- and post-conditions, entry and exit points, constrained regions, and non-functional properties on quality of service metrics etc. Based on this metamodel, a process modelling tool could suggest users possible and meaningful variants to compose the process fragments. In our future work we elaborate on this approach and its application in the business process execution language (BPEL), a standard for orchestration of Web services proposed by OASIS (2007). Nevertheless the approach is applicable to other graph-based process languages as well.

Acknowledgments
The work published in this article was partially funded by the COMPAS project[1] under the EU 7th Framework Programme Information and Communication Technologies.

References

Basel Committee on Banking Supervision (2006) International Convergence of Capital Measurement and Capital Standards: A Revised Framework, http://www.bis.org/publ/bcbs128.htm

Bernstein PA, Dayal U (1994) An Overview of Repository Technology. In: Proc. of the 20th Intl. Conference on Very Large Data Bases (VLDB), Morgan Kaufmann.

Bobrik R, Bauer T, Reichert M (2006) Proviado – Personalized and Configurable Visualizations of Business Processes. In: 7th Intl. Conference on E-Commerce and Web Technologies (EC-Web), Springer.

Caprasse D, Laurent J, Reed W (2008) Three Lines of Defence: How to take the Burden out of Compliance. In: European Insurance Digest April 2008, http://www.pwc.com/en_GX/gx/insurance/pdf/european_insurance_digest_april_2008.pdf

Daniel F, Casati F, D'Andrea V, Strauch S, Schumm D, Leymann F, Mulo E, Zdun U, Dustdar S, Sebahi S, de Marchi F, Hacid MS (2009) Business Compliance Governance in Service-Oriented Architectures. In: Proc. of the IEEE 23rd Intl. Conference on Advanced Information Networking and Applications (AINA'09), IEEE Press.

[1] http://www.compas-ict.eu/

Eberle H, Unger T, Leymann F (2009) Process Fragments. In: Proc. of the 17th Intl. Conference on Cooperative Information Systems (CoopIS), Springer.

European Project COMPAS (2009) State-of-the-art Report on the existing Approaches to Improving Reusability of Processes and Service Compositions. Project Deliverable D4.1, http://www.compas-ict.eu/results.php

Karastoyanova D, Leymann F (2009) BPEL'n'Aspects: Adapting Service Orchestration Logic. In: Proc. of the IEEE Intl. Conference on Web Services (ICWS), IEEE.

Kiczales G, Lamping J, Mendhekar A, Maeda C, Lopes CV, Loingtier JM, Irwin J (1997) Aspect-oriented Programming. In: Proc. of the European Conference on Object-Oriented Programming, Springer.

Küster JM, Ryndina K, Gall H (2007) Generation of Business Process Models for Object Life Cycle Compliance. In: Proc. of the 5th Intl. Conference on Business Process Management (BPM'07), Springer.

Leach P, Mealling M, Salz R (2005) A Universally Unique Identifier (UUID) urn Namespace, RFC4122.

Ly LT, Göser K, Rinderle-Ma S, Dadam P (2008) Compliance of Semantic Constraints – A Requirements Analysis for Process Management Systems. In: Proc. of the 1st Intl. Workshop on Governance, Risk and Compliance – Applications in Information Systems (GRCIS'08), CEUR proceedings.

Ma Z, Leymann F (2009) BPEL Fragments for Modularized Reuse in Modeling BPEL Processes. In: Proc. of the 5th Intl. Conference on Networking and Services (ICNS 2009), IEEE Computer Society.

OASIS (2007) Web Services Business Process Execution Language Version 2.0. OASIS Standard.

Oberortner E, Zdun U, Dustdar S (2008) Domain-specific Languages for Service-Oriented Architectures: An Explorative Study. In: Proc. of the 1st European Conference: Towards a Service-Based Internet (ServiceWave), Springer.

Sackmann S, Kähmer M (2008) ExPDT: A Policy-based Approach for Automating Compliance. In: Wirtschaftsinformatik (WI), 50(5), pp. 366-374, Springer.

Sadiq SW, Governatori G, Namiri K (2007) Modeling Control Objectives for Business Process Compliance. In: Proc. of the 5th Intl. Conference on Business Process Management (BPM'07), Springer.

Wolter C, Schaad A (2007) Modeling of Task-Based Authorization Constraints in BPMN. In: Proc. of the 5th Intl. Conference on Business Process Management (BPM'07), Springer.

[All links were last followed on November, 18th 2009]