Institute of Architecture of Application Systems

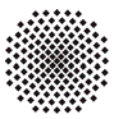
Fragmento: Advanced Process Fragment Library

David Schumm, Dimka Karastoyanova, Frank Leymann, Steve Strauch

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

BibTeX:

```
@inproceedings{SchummKLS10,
  author    = {David Schumm and Dimka Karastoyanova and Frank Leymann and
               Steve Strauch},
  title     = {Fragmento: Advanced Process Fragment Library},
  booktitle = {Proceedings of the 19th International Conference on Information
               Systems Development (ISD 2010), 25 August 2010, Prague,
               Czech Republic},
  year      = {2010},
  publisher = {Springer-Verlag}
}
```

Universität Stuttgart
Germany

Fragmento: Advanced Process Fragment Library

David Schumm, Dimka Karastoyanova, Frank Leymann, Steve Strauch

Abstract Reuse is a common discipline for decreasing software development time and for improving overall quality, independent from the domain. As business processes represent a fundamental asset of an organization, several concepts for enabling reuse during process modeling have been proposed. However, only few concrete examples for reusable process artifacts have been discussed so far. In this paper, we present the concept of process fragments and an example collection of process fragments for illustrating our reuse concept and for showing that it can actually be applied in practice for an easier and faster development of process-based applications. The fragment examples demonstrate different characteristics such fragments may exhibit. We also argue that this work will encourage reuse of process logic in terms of fragments, since it also provides an opportunity to design and develop a process fragment library for collecting process logic explicitly. As technical enabler for the approach we present a prototype called Fragmento.

1 Introduction

The current reusable granules in process design are language constructs like activities, control and data connectors, routing gateways, business rules, variables and other basic artifacts. The next larger and established reusability-related concept is sub-process, which already represents a self-contained and functionally complete unit for modeling and execution. Next in size are process templates, process variants and reference models, which are already the largest granular units for reuse and cover reusability and customizability for whole processes. Reuse of only a part or an artifact of a process is not covered by these approaches. We argue that within the range from basic language constructs to sub-processes and process templates there needs to be another, smaller unit of reuse which should allow fine-grained reuse of business logic. Process fragment presents a concept that

Institute of Architecture of Application Systems, University of Stuttgart,
Universitätsstraße 38, 70569 Stuttgart, Germany
{Schumm, Karastoyanova, Leymann, Strauch}@iaas.uni-stuttgart.de

fills this gap. The concept for reuse that we propose allows an easier and faster development of process-based applications. This includes for instance applications based on Web service compositions. Besides, it is a widely spread trend in software engineering to partially separate the application functions from the process logic orchestrating them.

This paper describes a collection of reusable building blocks for usage in process design based on the concept *process fragment*. In addition, we present an infrastructure component enabling the storage and management of fragments, which we call process fragment library. We discuss concrete examples of process fragments which we have identified during our research. The fragments have specific characteristics in which they differ from each other, for example the number of exits or if constraints are imposed on them. One objective of this work is to show the usefulness of the concept of process fragments by providing real examples that one can actually work with. In other disciplines, such as in traditional programming, code fragment libraries are a quite common source for reuse. Also in areas that are not related to computer science, libraries of reusable building blocks are widely used, e.g. in chemistry [12]. Within computer science such libraries are sometimes referred to as repository [1]. Various works on these special purpose databases exist for instance in the field of semantics in business processes [2] or agent systems [5]. In order to advance the state of the art we advocate the use of a repository providing advanced functionality for managing reusable process artifacts in different process languages. Additionally, we want to encourage the identification and publication of further process fragments in order to create an open library for capturing the progress of research and development in this field and see the presented work as technical enabler.

This paper is organized as follows: Section 2 describes a general concept for process fragments and exemplifies domain-specific extensions of this concept. Section 3 describes several process fragment examples. In Section 4, we present a process fragment library as supporting infrastructure for our approach. Related work on concrete process fragments is presented in Section 5. Finally, Section 6 summarizes the paper and identifies future work.

2 The Concept of Process Fragments for Reuse

In this section, we present briefly the concept of process fragments that is independent from the platform and technology that is chosen for the actual process language, implementation and serialization format for process fragments. In [19] we have given a general definition for the notion of process fragments. “A process fragment is defined as a connected graph 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 but is not required to: define a context (e.g., variables) and contain a process start or process end node. It may contain multiple incoming and outgoing 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 a way to complete it to an executable process graph. Therefore, a process fragment is not necessarily directly executable and it may be partially undefined. [19]" Depending on the particular language that is chosen for implementation, further characteristics are conceivable such as explicit data flow represented by data edges. Based on this definition we are able to express reusable pieces of process structure without limiting the expressiveness to single entry single exit (SESE) structures.

Depending on the particular application domain further requirements on the characteristics of a process fragment might be necessary. For instance, the focus in our research is set on managing compliance [19]. Compliance refers to all measures that need to be taken in order to adhere to laws, regulations and internal policies (corporate guidelines within the company). It is required that process fragments must not be changed to ensure the corresponding compliance feature. This means that the fragment may only be used the way it has been designed and only particular parts of it may be changed. This way it can be better ensured that after integration of a fragment into a process it still implements the compliance requirement that it has been designed for. For the usage of process fragments in the field of compliance we proposed in [19] additional characteristics: (i.) a process fragment may be parameterizable in order to mark points of variability which can render it abstract. The fragment is completed (i.e. concretized) when incorporated into a process; (ii.) the placeholders contained in a process fragment (i.e. regions) may be constrainable. By constraining the regions it can be defined how those placeholders may be filled with activities or other fragments.

Process fragments are reusable in process design in general and also in the field of (Web) service composition in particular. Apart from applying the fragments in modeling compositions they can be used to specify additional information like usage scenarios associated to services, compliance criteria a service meets etc.

3 Process Fragment Collection

We use the Business Process Modeling Notation (BPMN) [16] for representing process fragments graphically. We extended this notion with a cloud icon for representing a region. Parameters and constraints are expressed with an annotation icon that we created, see Figure 1. Entries and exits of a fragment are represented by control links that either have no target or no source. We use this notation in the scope of this work to ease understanding. The code fragment specified in the Business Process Execution Language (BPEL) [14] we discuss here does not make use of the extensions and can be represented with native language constructs.

As mentioned in the introduction, the fragments discussed in the following sections have specific characteristics in which they differ from each other. The first example fragment (approval) has multiple exits. The next fragment extends the fragment for approval with constraints and regions. The fragment realizing the "4-eyes principle" also uses constraints and contains a region for customization. A symmetric counterpart for usage in choreographies is provided by the fragment for secured interaction. The fragment for trusted timestamp exemplifies a domain-specific fragment for reuse. A particular control structure is implemented by the fragment for avoidance of infinite waits. These characteristics can be exploited as classification schema to support efficient search in the fragment library. The fragments of our collection have been manually identified in an industrial case study (compliance in a loan originating process) which has been defined by our project partner Thales Services SAS, France. For this paper we have selected rather simple fragments of this case study as they clearly illustrate the key concepts of our approach. After identification we have modeled the fragments in a process design tool and stored them in our process fragment library for later reuse in process modeling.

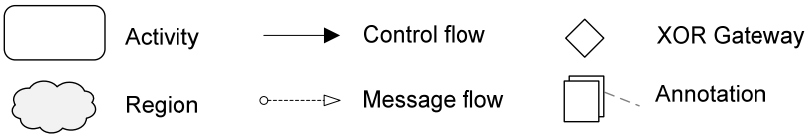


Fig.1. Process fragment constructs

3.1 Process Fragment for Approval

In many business processes and also in workflows (i.e. in the technical implementation of a process [11]) a step for checking a particular situation is required. For instance, for quality assurance there needs to be a check for mistakes and also for authorization reasons checks are necessary, as discussed in [19]. Typically, there are even multiple approvals within a single process, e.g. in approval chains. Figure 2 shows the process fragment for approval in BPMN. This fragment is applicable in almost any process language though. The fragment states that if a certain condition is met, a particular situation is assessed. This fragment has a single entry and in our design it has multiple exits, one for acceptance and the other for rejection. It has some characteristics which are likely to be parameterized: These are the activation conditions in which cases this approval needs to be performed, a staff query or a Web service for performing the check, and respective input values that should be approved, e.g. a document. Those parameters need to be set during concretization, i.e. during integration of the fragment into a process.

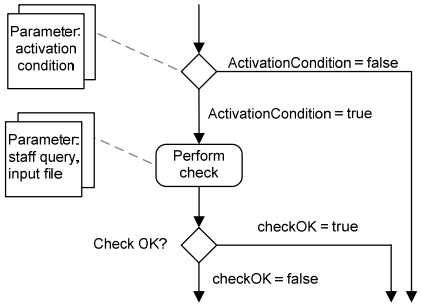


Fig.2. Process fragment for approval

3.2 Process Fragment for Approval with Constrained Region

In Section 2 we have remarked that specific extensions for process fragments depending on the application domain can be useful. In the field of compliance one required characteristic is that only particular parts of the fragment may be changed during integration into a process. For this we use an annotation mechanism for describing how particular parts may be changed during customization and integration of the corresponding fragment into a process, see Figure 3. To allow modification of the inner structure of a fragment in a controlled manner we propose to impose constraints on regions for compliant composition. In other scenarios regions could also be used without any constraints. In this example fragment the region allows integrating other steps in between the entry of the fragment and the approval step. However, disabling the approval must not be possible. Constrained regions could also be placed at the entries and exits of a fragment for enabling a constrained integration into a process.

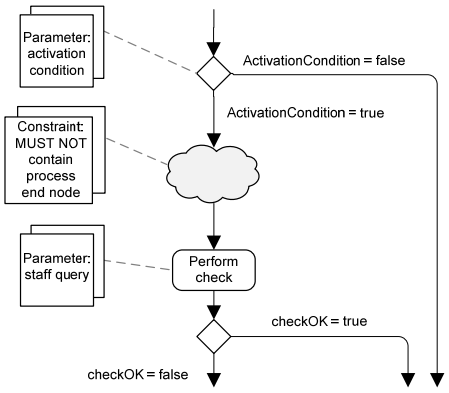


Fig.3. Process fragment for approval with constrained region

3.3 Process Fragment for 4-Eyes Principle

The 4-eyes principle (also called segregation of duty) is a frequent compliance requirement used for avoiding misuse and fraud, for security reasons or for avoidance of conflict of interest. For instance, in a banking application the customer requesting a loan and the clerk who may grant it must not be the same person. Typically, this requirement is realized using an annotation mechanism as the fragment in Figure 4 illustrates, combined with checking during runtime. The fragment in Figure 4 is designed for sequential execution. For parallel execution or for execution without control dependency other variants of this fragment need to be defined.

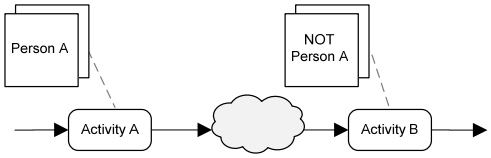


Fig.4. Process fragment for 4-eyes principle

3.4 Process Fragment for Secured Interaction

There are many different ways, methods and technologies for making an interaction with a process partner secure. This includes for instance transport layer security, message encryption and usage of signatures. The common way for integration of such functionalities is to annotate the activities which shall be executed in a secured manner. This annotation is interpreted and accordingly executed by the corresponding middleware. Nonetheless it can also be directly integrated into a process. Although the fragment shown in Figure 5 might only be used for documentation purposes and not be applied in process execution languages due to the before mentioned current practice, it is still an illustrative example for a fragment that has a corresponding counterpart. A counterpart in this context is another fragment designed for interaction and integration with the fragment from the partner's point of view. The number of counterparts depends on the particular interaction scenario. These kinds of fragments are important in Web-based application integration in which multiple processes and services need to interact with each other in a well-defined manner.

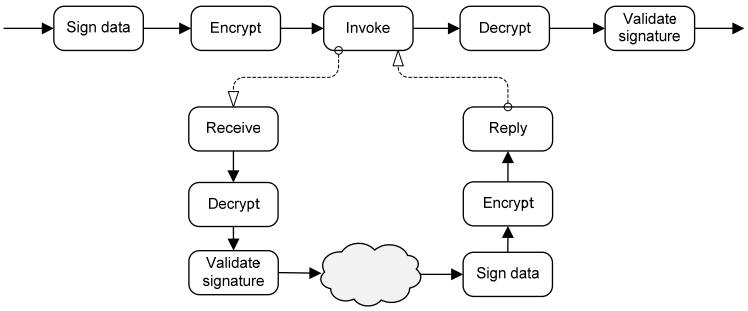


Fig.5. Process fragment for secured interaction (upper part) and the symmetric counterpart (lower part)

3.5 Process Fragment for Trusted Timestamp

For some business processes it is necessary to store a timestamp, e.g., when an offer is being sent out to a customer. For compliance reasons this timestamp needs to be "trusted" in particular cases, this means it has to be issued by a certified timestamp provider. Figure 6 shows a process fragment that has been designed according to the procedure for retrieving and validating a trusted timestamp defined in [13]. Basically, this fragment could be used for integration of trusted timestamps into a process without requiring in-depth knowledge on the details of the procedure. Possibly, this fragment could even be offered from the timestamp provider for easier and faster integration with the offered (Web) services. We argue that process fragments can be used as an annotation to a service (or process) for providing additional meta-information about it, going beyond the description of its interfaces and usage policies. We consider this approach in our process fragment library. Please note that this fragment could also be implemented as a sub-process, however with limited customization capabilities.

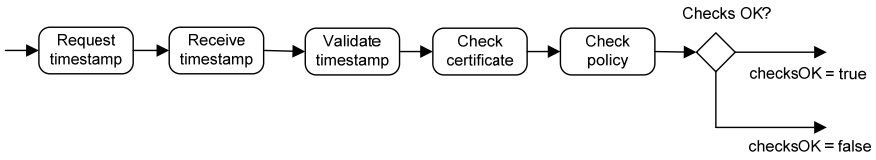


Fig.6. Process fragment for trusted timestamp

3.6 Process Fragment for Avoidance of Infinite Waits

Process fragments are concrete solutions to frequently occurring, but also to specific problems. The process fragment for avoidance of infinite waits (see Listing 1) implements a control structure in BPEL which takes care that a process does not hang up in case a service which has been invoked does not respond. This can be achieved by using a `<pick>` construct in combination with the receiving activity `<onMessage>` that awaits the response. If the response is not received in time, the `<onAlarm>` construct registers a timeout and cancels waiting for the message and thereby prevents the process from hanging up. This control structure is not really complex, but in case a process designer is not sure how to deal with this problem it becomes quite handy. Even if the designer knows how to model this, reusing this fragment can at least speed up overall development time. Another fragment defining control structures is best-practice in process design: for dynamically changing endpoint references of service invocations during runtime an `<eventHandler>` construct with a nested `<assign>` activity can be used.

Listing 1. Process fragment for avoidance of infinite waits

```
<sequence name="main">
   <invoke name="invokeService" .../>
   <pick name="pick">
        <onMessage ...>
          <assign name="assignResponse" validate="no" />
        </onMessage>
        <onAlarm for="P1DT00H">
          <assign name="assignTimeoutOccurred" />
        </onAlarm>
   </pick>
</sequence>
```

4 Supporting Infrastructure

In this section we present the process fragment library which is the special purpose component for storage and management of process fragments. We have developed a prototype of such a library, called Fragmento [17], its conceptual architecture is presented in Figure 7. Fragmento is dedicated to the management of BPEL processes, WSDL documents, WS-Policy Annotations, especially BPEL process fragments and other process-related artifacts.

Beyond the basic functionality for management of versions, locks and (typed) relations we have implemented several functions which are helpful particularly in the management of processes and process fragments, see Figure 7. For example, the basic search functionality operates on fragment meta-data, like the fragment name, keywords, the number of entries and exits, in which domain it is used and other classifications (currently full text search). In addition, Fragmento provides

an extensibility mechanism for integration of custom query functions. This allows the implementation of search functions beyond the meta-data of an artifact, e.g. concerning the structure of a fragment or related to properties of its annotations.

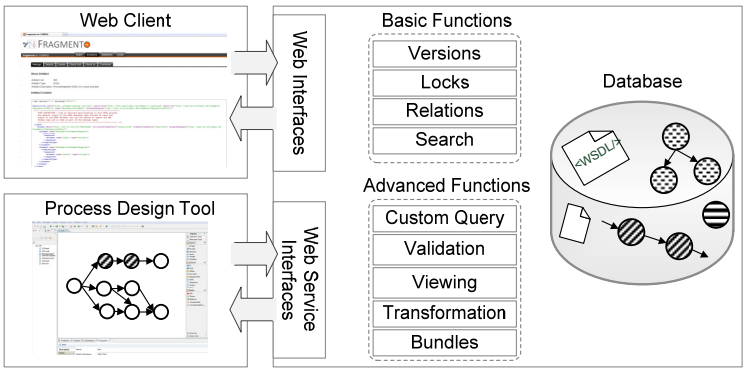


Fig.7. Conceptual architecture of Fragmento

Process design and process enactment require valid models for proper execution. Thus, Fragmento also provides XML schema validation and an extensibility mechanism for additional validation functions that could be used to check if a process model contains cycles for example. For flexibly creating user-specific representations and variants of processes and process fragments on the fly, Fragmento supports process view transformations [18]. These transformations include for instance the omission of attributes and activities that match particular characteristics (e.g. for removal of activities related to debugging). Furthermore we have integrated a transformation that changes language extensions used in a process fragment (e.g. for representing regions) into standard constructs for compatibility with other tools. We consider the mechanisms for easy retrieval of process fragment information a valuable feature and we therefore support the definition of bundles, which enables packaging all artifacts related to a process (or fragment) into one SOAP message (or Web page).

Fragmento exposes all of its offered functionality as Web service (currently via SOAP/HTTP binding). It also provides a Web-based interface to allow direct access to the repository over the Web. For the Web client we use double tab navigation. On top level the user can choose between the management of artifacts, relations or locks. On the second level the particular management functions for the corresponding top level selection are shown. The integration with a process design tool based on Eclipse is part of our current research agenda.

Fragmento extends an existing repository code base that has been developed by the MASTER project [20]. The technology stack for Fragmento consists of a Tomcat application server which is hosting the repository application. Hibernate is used as data abstraction layer. Furthermore, the Spring Framework is employed for object lifecycle management and a PostgreSQL database is utilized for storage.

For the development of the Web service interfaces Axis 2 libraries have been used. The Web client is built using Java Server Pages (JSP) and Tag Libraries for the view, while Servlets are used as controller for handling client requests. Fragmento is a Java application. All the fragments presented in this work can be stored and managed by Fragmento. More documentation of the functionality, provided interfaces, screenshots and implementation details can be found in [17].

5 Related Work

In the following we discuss work related to concrete process fragments. A comparison of our concept to approaches on process reuse in general and a discussion of the life cycle of process fragments can be found in our former work [19]. Notable for this paper are the results of the ProWAP project. With the term Workflow Activity Pattern the authors refer to the description of recurrent business functions that are frequently used in business processes. In [21] a set of seven activity patterns based on literature study is discussed. These activity patterns are namely Approval, Question-answer, Uni-/Bi-directional Performative, Information Request, Notification and Decision Making. In this work the activity patterns are defined as SESE fragments (without placeholders), similar to sub-processes. This limitation intends to ease pattern implementation, pattern reuse and pattern composition within process design tools. We see the patterns discussed in [21] as an additional source of concrete fragments, however the fragments we presented and the patterns discussed in the mentioned article are just the tip of the iceberg.

We would like to stress that we see a difference between the terms pattern and fragment. For instance, in [3] Workflow Patterns are discussed. Workflow Patterns describe elementary language constructs which are supported by workflow languages, for example sequential execution, parallel split or exclusive choice. The expressiveness of workflow languages differs, thus some workflow patterns might be supported by a particular language while some others might not. The patterns described in [3] are somehow reusable building blocks and [7] even shows that these workflow patterns can in fact be applied as modeling granules for accelerating process development. However, a pattern is an abstract solution concept to frequent problems while a fragment is a more concrete solution, possibly to a quite specific problem. A fragment could more or less be compared to a code snippet, while a pattern is more conceptual, like a design pattern in terms of [6]. Another example for patterns in this context is Message Exchange Patterns (MEP) [9]. According to [4], MEPs define the sequence, the cardinality, the source and the recipient of messages. For instance, Request-Response is such a pattern. These patterns can also be applied in process design in the way shown in [7], but still they are quite abstract forms of reusable building blocks.

An approach that provides patterns that enhance the reliability of a BPEL process is shown in [10]. The work makes use of a guideline for defining reusable

fault handling logic [15] and discusses four abstract solution patterns and BPEL code fragments for fault-tolerant service invocation. The authors propose to annotate the reliability requirements to the process and use a model transformation to automatically integrate the fragments accordingly. In summary, the fragments described in [10] are domain-specific, concrete, language-specific and, which is most important here, they are reusable and useful for making process design easier and faster.

6 Conclusion and Future Work

In this paper we have presented our approach of process fragments for reuse in process design. The main contribution of this paper is a collection of concrete process fragments which illustrates that there is a need for this concept and that it actually can be applied in practice. The fragment examples we presented are of a rather simple nature in order to clearly illustrate the key concepts of the approach. As technical enabler we presented a process library prototype called Fragmento. The architecture of the prototype and its functionality were discussed in detail. The process library supports storage and management of recurring and reusable process fragments without focusing on a particular application domain.

In many different fields, for instance in grid computing, manufacturing workflows or scientific workflows, there are most likely domain-specific and language-specific but also general process fragments which can be manually identified and subsequently reused. Fragments from particular application domains may also be useful in other domains, or bring up new ideas which are helpful in many fields. In [8] techniques for fragment discovery in the field of scientific workflows have been proposed, they are basically also applicable for fragment discovery in a business context. Furthermore the case study evaluation in [8] states that there is definitely a need for workflow fragments and reusable service composition in e-Science scenarios.

At present, we are investigating methods and limitations of translating fragments representations from one process language into another. We are also investigating techniques for isolating and extracting reusable process fragments from existing processes. In our future research we will work on a classification of the different forms and characteristics of process fragments. We are convinced that diversity in research on fragments will be beneficial for the further development of the overall fundamentals, concepts and related techniques. The presented collection of process fragments can be seen as a starting point for future research concerning reusable building blocks of process logic.

Acknowledgments. The work published in this paper was partially funded by the COMPAS project (www.compas-ict.eu) under the EU 7th Framework Programme Information and Communication Technologies (contract no. FP7-215175) and the S-Cube project (www.s-cube-network.eu) under the Network of Excellence (contract no. FP7-215483).

References

1. P. Bernstein, U. Dayal: An Overview of Repository Technology. Proc. of the 20th International Conference on Very Large Data Bases (VLDB), Morgan Kaufmann, 1994.
2. Z. Ma, B. Wetzstein, D. Anicic, S. Heymans, F. Leymann: Semantic Business Process Repository. Proc. of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM), 2007.
3. W. van der Aalst, A. ter Hofstede, B. Kiepuszewski, A. Barros: Workflow Patterns. In: Distributed and Parallel Databases, 14(1):5–51, Springer, 2003.
4. J. Nitzsche, T. van Lessen, F. Leymann: WSDL 2.0 Message Exchange Patterns: Limitations and Opportunities. Proc. of the 3rd International Conference on Internet and Web Applications and Services (ICIW), IEEE, 2008.
5. V. Seidita, M. Cossentino, S. Gaglio: A Repository of Fragments for Agent Systems Design. Proc. of the Workshop on Objects and Agents (WOA06), 2006.
6. E. Gamma, R. Helm, R. Johnson: Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley Longman, 1995.
7. T. Gschwind, J. Koehler, J. Wong: Applying Patterns During Business Process Modeling. Proc. of the 6th International Conference on Business Process Management (BPM), Springer, 2008.
8. A. Goderis, U. Sattler, P. Lord, C. Goble: Seven Bottlenecks to Workflow Reuse and Repurposing. Proc. of the 4th International Semantic Web Conference, Springer, 2005.
9. W3C: Web Services Description Language (WSDL) Version 2.0: Additional MEPs, W3C Working Group Note, June 2007.
10. A. Liu, Q. Li, L. Huang, M. Xiao: A Declarative Approach to Enhancing the Reliability of BPEL Processes, IEEE International Conference on Web Services (ICWS), IEEE Computer Society, 2007.
11. F. Leymann, D. Roller: Production Workflow: Concepts and Techniques. Prentice Hall PTR, 2000.
12. Maybridge: The Maybridge Ro3 Fragment Library, product flyer, 2007. http://www.maybridge.com/images/pdfs/ro3frag.pdf
13. Network Working Group: Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP), RFC 3161, 2001. http://tools.ietf.org/html/rfc3161
14. Organization for the Advancement of Structured Information Standards (OASIS): Business Process Execution Language 2.0 (BPEL). 2007.
15. L. Zeng, H. Lei, J. Jeng, J. Chung, B. Benatallah: Policy-Driven Exception-Management for Composite Web Services. Proc. of the 7th IEEE International Conference on E-Commerce Technology (CEC), IEEE Computer Society, 2005.
16. Object Management Group: Business Process Modeling Notation (BPMN), OMG Available Specification, Version 1.1, January 2008.
17. Fragmento - Fragment-oriented Repository. Online Documentation, 2010. http://www.iaas.uni-stuttgart.de/forschung/projects/fragmento/start.htm
18. D. Schumm, F. Leymann, A. Streule: Process Viewing Patterns. Proc. of the 14th IEEE International EDOC Conference (EDOC 2010), IEEE Computer Society Press, 2010.
19. D. Schumm, F. Leymann, Z. Ma, T. Scheibler, S. Strauch: Integrating Compliance into Business Processes: Process Fragments as Reusable Compliance Controls. Proc. of the Multikonferenz Wirtschaftsinformatik (MKWI'10), 2010.
20. European Project MASTER: Technical Architecture and APIs for Single Trust Domain. Project Deliverable D2.3.1, 2009. http://www.master-fp7.eu
21. L. Thom, M. Reichert, C. Iochpe: Activity Patterns in Process-aware Information Systems: Basic Concepts and Empirical Evidence. In: International Journal of Business Process Integration and Management (IJBPIM), 2009.