Implementation Architectures for Adaptive Workflow Management

Hanna Eberle, Frank Leymann and Tobias Unger
Institute of Architecture of Application Systems (IAAS)
University of Stuttgart
Stuttgart, Germany
email: eberle@iaas.uni-stuttgart.de, leymann@iaas.uni-stuttgart.de, unger@iaas.uni-stuttgart.de

***Abstract*—Business processes are often required to be highly flexible and adaptive due to the fact, that business conditions change. Therefore, there exist a lot adaptation and flexibility concepts for workflows. However, workflow adaptation concepts are often discussed on the language level neglecting a discussion on the implementation architectures. Until now, effective implementation architectures have not been investigated. Therefore, the main contribution of this work is to research three implementation strategies for adaptive workflow management, which we discuss with respect to modeling requirements and change management.**

***Keywords*-Adaptable Workflows, Flexible Workflows, Workflow Execution Architecture**

I. Introduction

Adaptation concepts in the workflow technology are manifold. Since business conditions change, business processes are often required to be highly flexible and adaptive. Workflow execution flexibility is a key enabler for workflow adaptation, because if a system does not provide any means for flexibility the system cannot adapt. In the workflow domain, the terms workflow flexibility and workflow adaptation are often used synonymously. In this work we also discuss workflow adaptation from the flexibility point of view. Furthermore, we distinguish between workflow adaptation and workflow evolution. The changes of the business logic become visible to one instance only in workflow adaptation, while the change of a workflow model in terms of workflow evolution affects all running instances. Traditional workflow languages provide means to specify partial orderings between units-of-work, which are also called activities. Most workflow languages rely on graph theory [12] and define the partial orderings between the activities as directed graphs, where the edges in these workflow graphs are called control connectors. Today, applications are often realized as Web Services following the SOA paradigm and workflows orchestrate the different Web Service applications to become a new and more complex application. Therefore, activities send and receive messages to interact with the activity implementations represented as Web Service [18]. Execution flexibility in workflows can be established in two ways, either the partial ordering of activities or the set of activities is adapted, which is also called adaptation of business logic, or the selection and binding of activity implementations is performed at runtime and therefore provides a point of flexibility. In this paper we focus on the changes of business logic.

The state-of-the-art in the domain of business logic adaptation of workflows, however, focuses on the change and change management of workflow models on a language and conceptional level, defining business logic change operations and investigation on correctness criteria for the correct changes for a certain set of instances [14] [2].

Until now, effective implementation architectures have not been investigated. Most adaptation approaches such as process fragments [5] are implemented using instance migration c.f., [4]. However, instance migration is not a sufficient implementation approach for the process fragment-based adaptation approach as it requires e.g. to suspend the process execution while the migration. Other adaptation implementations for adaptive workflows are feasible, which are implementing the process-fragment adaptation concept more natively and overcome some disadvantages of the instance migration implementation approach. Therefore, the aim of this work is to research and discuss new implementation architectures for adaptive workflow management.

We present three adaptation implementation architectures for adaptive workflow management of internal workflow artifact adaptation and discuss these approaches with respect to modeling requirements and change management.

The paper is organized as follows. First, in Section IV, we discuss the related work runtime adaptation concepts in the workflow domain for graph-based workflows. We examine the workflow adaptation concepts from a modeling language perspective and the implementation concepts subsequently, that are available to implement the workflow adaptation concepts. Secondly, we present and discuss adaptation implementation architectures, which realize one adaptation implementation concept in Section V. The adaptation implementation architectures handle adaptation internally by adapting the internal model artifacts. We conclude our work in Section VI.

II. Traditional Workflow Execution

A. Execution Architecture

Workflow languages are programming languages that are interpreted, like e.g., Java [6]. Source code, written in interpreter programming languages, are not complied into machine

 ISBN: 978-1-61208-109-0

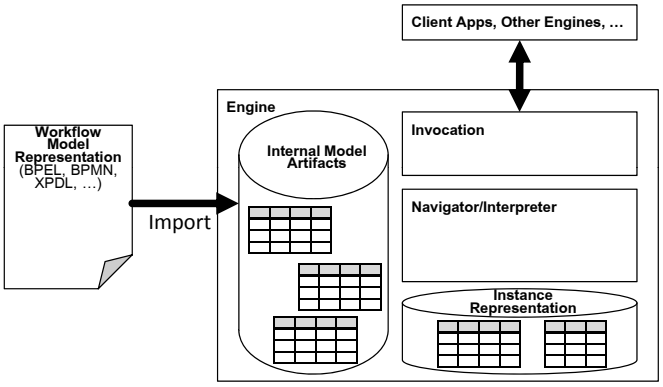

Figure 1: WFMS Architecture (c.f. [7])

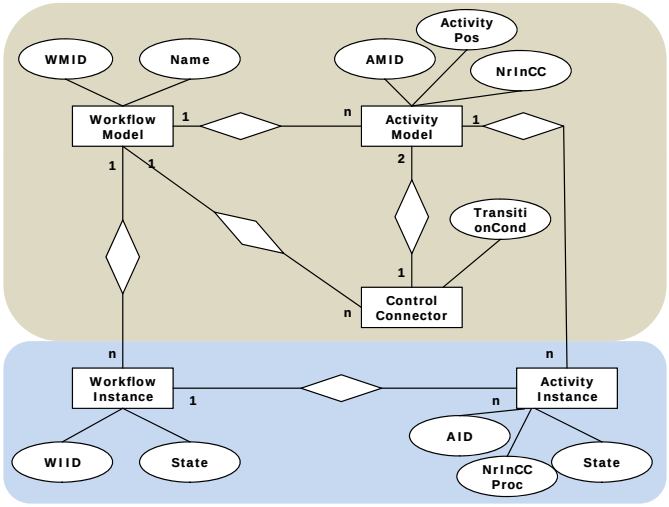

Figure 2: ER-Diagram (c.f. [12])

code, which can be directly run by a machine without any other additional components or programs. The source code must be complied into the platform specific source code. Once it is complied it cannot be run on another platform, but must be complied using another platform specific complier. Source code of interpreter languages are translated into a representation, e.g., byte-code, which is run by a interpreter or a virtual-machine, where the interpreter is platform specific, but the source code representation is platform independent. Workflow models act like source code, which get imported and deployed to a virtual-machine for workflow execution, which is widely called Workflow Management System (WfMS). A simplified WfMS architecture based on [7] [12] is presented in Figure 1. Workflow models are represented in a workflow language, like the XML based language BPEL [13]. At deployment time the workflow models are imported and translated into internal artifacts [11] (c.f. 2), e.g., table entries of a relational representation [12] or Java objects. The internal artifacts are interpreted by the Navigator Component. In particular, workflow languages are designed to prescribe orderings between activities. Activities represent business activities, basically, where the activity implementation of an activity might be a client application implemented in another programming language or even implemented by another process. The sending and receiving of messages is handled by the Invocation Component. Workflow models are imported, activated and deactivated using workflow model management API functions. The workflow instance API functions manage the state of a workflow instance, e.g., to suspend and resume a workflow instance as well as a function to terminate a workflow instance.

B. Internal Representation

A simplified schema for the relational representation of workflow models is presented in Figure 2. A workflow model consists of a set of activity models, which are connected using control connectors. An activity model is aware, whether it is a start or an end activity of the workflow, which is stored in the ActivityPos attribute. The NrInCC attribute represents the amount of incoming control connectors. A control connector can be annotated by a transition condition, which is a function mapping to a Boolean value. Each process model might have more than one concurrently running process instance. Workflow instances are represented by the state of the workflow instance and the states of its activity instances. The activity instance attribute NrInCCProc keeps track with the amount of already evaluated incoming control connectors.

C. Navigation

A workflow instance is created applying one of the instantiation patterns as presented in [3], e.g., a message or an event is received at an endpoint of a start activity. The Navigator uses the relational representation of the workflow model and the state of the workflow instance to determine the next activity to be executed. An activity is executed, if all incoming control connectors are evaluated. The activity instance keeps track with the amount of evaluated control connectors by increasing the amount of the attribute NrInCCProc by one each time the activity is target of an evaluated control connector. After an activity is finished the associated outgoing control connectors are selected, evaluated. A workflow instance completes successfully, if all end activities are finished.

III. Adaptation Example

To provide a more concrete idea of the business logic adaptation, we present a little example on what shall be achieved by business logic adaptation. The adaptation of the business logic in workflow models influence the produced execution traces of the workflow instances. An execution trace is denoted by the execution ordering between the activity instances. Our example workflow model consists

 ISBN: 978-1-61208-109-0

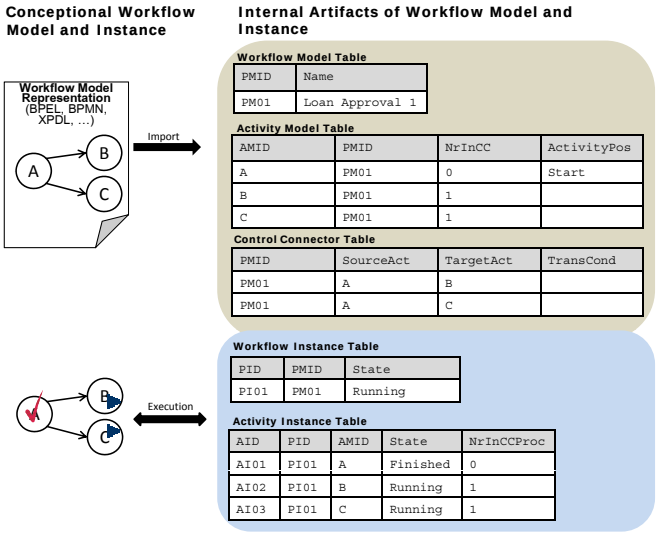

Figure 3: An Example

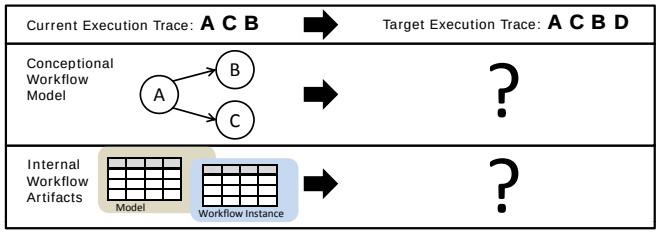

Figure 4: The Problem

of three activities A, B and C. A workflow instance of the workflow model has two running activities, B and C. A possible conceptional and internal representation of the example is depicted in Figure 3. The execution trace of that instance is *ACB*. The target execution trace of the adaptation is *ACBD*. An activity D shall be executed additionally, with the constraint that the activity D must be executed after the completed execution of activity C. Two questions arise from this little example of workflow adaptation. How is the workflow model modeled and adapted conceptionally, and how is the workflow model adapted internally? These questions are discussed in the remainder of this paper.

IV. Related Work

Adaptation approaches are manifold in the workflow domain. To be able to classify the approaches we investigate the approaches by regarding two aspects: (i) How is the adaptation achieved from a modeling point of view? (ii) What are the possible implementation strategies for these adaptation concepts?

A. Modeling Adaptive Workflows

There exist two basic adaptation approaches for graph-based workflow modeling languages. These approaches denote the ends of a spectrum. Hybrid approaches are possible. The first adaptation approach acts on the assumption, that a complete process model is modeled at design time. At runtime the process model needs to be adapted, due to the fact, that e.g., the business conditions might have changed. Activities are added or removed by the adaptation or the activity ordering is changed. The changes in the model are either applied to all instances of the process model, or made visible to one instance only. Approaches, that pursue that modeling approach, are presented in [2] [15]. The second adaptation approach acts on the assumption, that parts of a comprehensive process model are known at modeling time, e.g., by different parties. These parties are able to model the parts of a comprehensive process model. The comprehensive process model is assembled of these process parts. The assembling may currently available knowledge in terms of context data into account. That way the comprehensive process model is created adaptively, as the selection of the to be integrated parts and the integration of the different parts can be based on adaptation relevant data. The comprehensive process model creation might be performed either at runtime or at design time. If the selection and integration is performed at runtime, even runtime data can be employed by the selection. There exist various modeling approaches for the process part modelling, e.g., such as subprocesses [10], worklets [17], proclets [1] [16], process fragments [9].

B. Adaptation Implementation Strategies

In this section, we discuss the possible implementation strategies for the adaptation concepts. Afterwards, we investigate, which conceptional adaptation concept can be implemented by which of the implementation concepts, since not every adaptation concept can be implemented employing every implementation concept. We could identify two different adaptation management strategies for adaptation of graph-based workflows. The first implementation approach is to manage adaptation of the conceptional workflow model as a change of the internal model. The change of the workflow model is mapped to the internal artifacts as well. This strategy can be applied to both adaptation modeling approaches presented previously. This approach needs to be aware, whether the change shall be applied to one instance only or to all instances running on that model. In the case a model is adapted the change must comply to the execution traces of the instances, which are about to be adapted and run on the new conceptional model. Both adaptation modeling approaches can be implemented using this implementation strategy. Implementation architectures for this adaptation implementation strategy are discussed in subsequent sections of this paper. The second adaptation implementation strategy does not change the internal representation of the model at all. The adaptation of the workflow model, e.g., the composition of a part of a process with another part of a process is

 ISBN: 978-1-61208-109-0

	Conceptional change of Model	AOP	Process Composition
Change of Internal Model	✓	✓	✓
No Change of Internal Model Map to Interactions		✓	(✓)

Figure 5: Overview

mapped to interactions between the processes, like e.g., subprocesses, where the subprocess is integrated into the parent process by sending messages to the subprocess, which might be running in a different process context. However, the interaction must be implemented by a complex coordination protocol to be able to make the interacting processes appear to the outside as one process. The interaction also requires the interaction points to be modeled. This approach is not applicable, if the changes of the conceptional model are not modeled as a separate model or if parts of a process part need to be adapted and the interaction points are already modeled in the to be integrated parts. Changes in the pre-modeled parts of a process model can only be implemented by an implementation architecture, which is able to change the corresponding internal artifacts as well. Whereas the composition of pre-modeled parts can sometimes be mapped to interaction relations or always by implemented by changing the internal artifacts of the internal model adding new control connectors to glue the pre-modeled parts together. Whether the interaction approach can be applied, depends on the pre-modeled process parts and the integration points and operations, that are provided by the process part. The integration approach in not that flexible as the internal change or model approach, since the points of integration must be pre-modeled. The summary of the discussion is presented in Figure 5.

V. Adaptation Architectures applying the Internal Changes Strategy

In the following, we introduce adaptation architectures, where the conceptional changes of the workflow model are mapped onto changes of the internal workflow representation. We do not answer the question, whether an adaptation step is correct for an instance, which is discussed in various papers, e.g., in [2], [14]. We discuss implementation architectures for adaptation. The first implementation architecture supports the instance migration concept as presented in WebSphere MQ Workflow [8]. Instance migration associates an instance of a workflow model with another workflow model. The second implementation architecture supports the adaptation of a process model natively. The third and last approach presented in this paper supports the composition of pre-modeled process parts natively. We base our discussion on the internal workflow model and instance representations as presented in Section II.

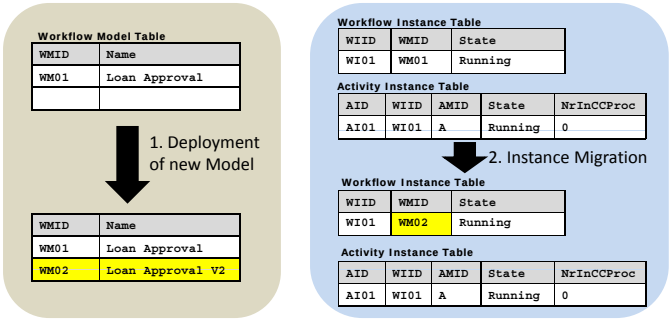

Figure 6: Instance Migration

A. Adaptation Architecture supporting Instance Migration

Instance migration associates a suspended workflow instance with another workflow model, where the target workflow model must fit to the already executed parts of the instance, and resumes the execution. Adaptation approaches applying instance migration are discussed e.g., in [2]. The concept of instance migration can be applied to implement adaptation, thus adaptation can be mapped instance migration, because the old workflow model can be changed and imported as new workflow model, which becomes the new workflow model of the instance. The internal artifacts are influenced by the migration as depicted in Figure 6. The workflow instance identifier, WIID, of the instance keeps the same, but the workflow model identifier, WMID, is changed to the WMID of the new workflow model. The traditional APIs can be used to import the new workflow model to the WfMS as well as to suspend and resume the workflow instance. A new migration API function needs to be added to be able to select and migrate a specific instance to a new workflow model. The instance migration approach allows other workflow instances associated with the old model. Since only the selected instance is migrated and the old workflow model is not changed, the remaining instances can finish on the old workflow model. This approach is suitable, if the adaptation of a workflow instance is an exceptional case. The adaptation is percepted to the outside, since the workflow instance is now associated with another workflow model. However, message correlation issues must not be handled differently, since the correlation is based on business case data, which is associated with the instance.

The perception of the process now might be different from the one perception before the adaption step, since it might be possible that new activities are deployed, which receive and send messages.

B. Adaptation Architecture supporting Changes of the Internal Representation

Another implementation approach for adaptable workflows is to change the workflow model directly, both the workflow model and the internal representation as presented in Figure

 ISBN: 978-1-61208-109-0

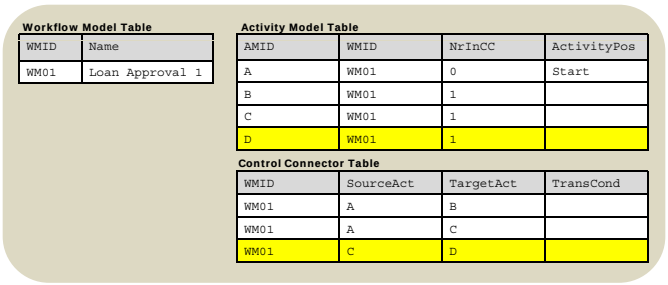

Figure 7: Extending the Model Example

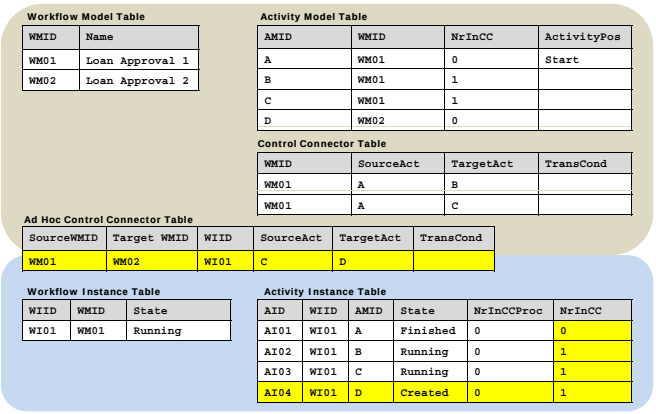

Figure 8: Connecting two Models by the Definition of an Ad Hoc Control Connector

7. The workflow model is extended by the adaptation logic. The new activities are deployed and the new activities and control connectors are written into the workflow model tables. This approach has the advantage, that only the workflow model is changed as depicted in Figure 7, while nothing needs to be changed at the instance level. This approach is only applicable, if all instances running on the changing model comply to the change. This approach is suitable, if a workflow model needs to be changed very often and the workflow model has only very few instances running on it. This approach suits best for late modeling approaches and ad hoc workflows, if the relationship between model and instance is one-to-one, where the repetition rate is very low and the workflow model must be highly flexible. The same new API function needs to be added to the API function set as for the instance migration. Input to this function is the workflow model ID of the workflow model to be changed and the new workflow model. Internally, the function defines the new parts of the workflow model and deploys these parts and maps them to the internal representation to extend the workflow model tables.

C. Adaptation Architecture using Inter-Workflow-Model Control Connectors

If the target workflow model of the adaptation is a composition of already deployed workflow parts, which are already represented internally, we are able to map the target conceptional workflow model internally by introducing ad hoc control connectors. These control connectors connect activities of two different workflow models. However, this control connector is visible to one instance only, since the composition is only valid for one instance. The ad hoc control connector is hybrid, since it represents model information, but is only evaluated by one instance. This idea of this approach is quite close to the implementation idea of the interacting implementation strategies. However, no complicated coordination protocols must be executed to coordinate the executions of the different parts of the overall process, because the instance is not distributed over many execution contexts. The whole instance is executed within the same execution context and the complete workflow model is represented internally by the insertion of ad hoc control connectors as presented in Figure 8. The insertion of the ad hoc control connector changes the amount of incoming control connectors of the target activity. Therefore, we extend the activity instance table by a NrInCC column, which represents the amount of common incoming control connectors plus the amount of incoming ad hoc control connectors. The activity instances of a workflow part are created after this workflow part has been integrated into the workflow and the NrInCC is set during creation. A standard navigator must be adopted to be able to navigate over these workflow models and ad hoc control connectors. Therefore we first navigate the standard control connectors of the source activity afterwards we navigate over the ad hoc control connectors. Since all process parts are pre-modeled, the pre-modeled parts do not change and therefore the perception of the parts and interaction possibilities do not change. The most extreme case in this realization is to model all control connectors as inter process control connectors and each activity itself is a process model. Also this approach has its restrictions, since e.g., activities cannot be removed.

D. Discussion

The adaptation implementation architectures are powerful adaptation implementation strategies. These adaptation architectures enable to adapt a executed workflow model either by adding or removing parts of the workflow model. However, each of the implementation strategies is optimized towards a specific modeling paradigm and application scenario. The instance migration strategy was introduced first to manage versioning issues. Instance migration requires a smart model management due to the fact, that the amount of very similar workflow models grows. Instance migration requires the process instances to be suspended, before the instances can be migrated to the new version of the process model. The instance migration implementation concept is especially suitable to handle exceptional cases. The happy path is modeled in the original process model and in case a fault

 ISBN: 978-1-61208-109-0

occurs the faulting instance is drawn onto another process model, which is able to deal with the faulting situation. Faults are usually unexpected and therefore its not possible to cover all these faults by a process model. Whereas the native adaptation support implementation architecture should be applied for workflows, which are known to be adapted in an ad-hoc and therefore only one instance is created per natively adapting workflow model. The instance does not need to be suspended to perform the adaptation of the workflow model. Both of these approaches, the instance migration and the native ad-hoc changes of the model, change the set deployed activities and therefore might also change the set of activity message endpoints. Therefore the perception of the workflows are changed, since they now provide more endpoints, that can be called. The third implementation architecture is optimized and tailored towards adaptation concepts similar to the process fragment concept. The local knowledge is modeled locally and deployed to an engine. Depending on the runtime context data the parts are glued together by introducing instance control connectors. The process instance needs not to be suspended and no new activities need to be deployed. The perception of the process fragments does not change to the outside.

VI. Conclusion

In this work, we analyzed modeling and implementation approaches, which realize adaptation of business logic of workflows. We presented three implementation architectures for adaptive workflow management. The adaptation is handled by changing the executed model internally. We discussed the applicability of each of these implementation architectures and we argued, that the applicability and suitability of the implementation strategy chosen depends on the scenario that shall be implemented. The scenario specifies e.g., what information is known at modeling time, or what information is pre-modeled. In our future work we will evaluate the concepts as presented in this paper implementing all the approaches and compare these approaches with the already implemented instance migration approach.

Acknowledgment

This work is partially funded by the ALLOW project. ALLOW (`http://www.allow-project.eu/`) is part of the EU 7th Framework Programme (contract no. FP7-213339).

References

[1] M. Adams, A. H. M. ter Hofstede, D. Edmond, and W. M. P. van der Aalst. Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows. In R. Meersman and Z. Tari, editors, *OTM Conferences (1)*, volume 4275 of *Lecture Notes in Computer Science*, pages 291–308. Springer, 2006.

[2] F. Casati, S. Ilnicki, L.-j. Jin, V. Krishnamoorthy, and M.-C. Shan. Adaptive and Dynamic Service Composition in eFlow. In *CAiSE '00*, pages 13–31, London, UK, 2000. Springer.

[3] G. Decker and J. Mendling. Instantiation Semantics for Process Models. In *BPM '08*, pages 164–179, Berlin, Heidelberg, 2008. Springer.

[4] H. Eberle and S. Telezhnikov. *JBPM Fragment Engine*, 2010. http://code.google.com/p/jbpm-fragment-engine/.

[5] H. Eberle, T. Unger, and F. Leymann. Process Fragments. In *Proceedings CoopIS 2009*, pages 398–405, Berlin, Heidelberg, 2009. Springer.

[6] J. Gosling, B. Joy, and J. S. Guy L. *The Java(tm) Language Specification*. Addison-Wesley Longman, Amsterdam, June 2005.

[7] D. Hollingsworth. Workflow Management Coalition - The Workflow Reference Model. Technical report, Workflow Management Coalition, Jan. 1995.

[8] IBM Corporation. *IBM MQSeries Workflow: Concepts and Architecture*, 2006.

[9] K.-H. Kim, J.-K. Won, and C.-M. Kim. A Fragment-Driven Process Modeling Methodology. In O. Gervasi, M. L. Gavrilova, V. Kumar, A. Laganà, H. P. Lee, Y. Mun, D. Taniar, and C. J. K. Tan, editors, *ICCSA (3)*, volume 3482 of *Lecture Notes in Computer Science*, pages 817–826. Springer, 2005.

[10] O. Kopp, H. Eberle, F. Leymann, and T. Unger. The Subprocess Spectrum. In *Proceedings of the Business Process and Services Computing Conference: BPSC 2010*, Lecture Notes in Informatics. Gesellschaft für Informatik e.V. (GI), September 2010.

[11] F. Leymann. BPEL vs. BPMN 2.0: Should You Care? In *2nd International Workshop on BPMN*, Lecture Notes in Business Information Processing. Springer, October 2010.

[12] F. Leymann and D. Roller. *Production Workflow - Concepts and Techniques*. PTR Prentice Hall, January 2000.

[13] OASIS. Web Services Business Process Execution Language Version 2.0. Committee specification, OASIS Web Services Business Process Execution Language (WSBPEL) TC, Jan. 2007.

[14] M. Reichert and P. Dadam. $ADEPT_{flex}$-Supporting Dynamic Changes of Workflows Without Losing Control. *J. Intell. Inf. Syst.*, 10(2):93–129, 1998.

[15] M. Reichert, S. Rinderle-Ma, and P. Dadam. Flexibility in process-aware information systems. *T. Petri Nets and Other Models of Concurrency*, 2:115–135, 2009.

[16] W. van der Aalst, P. Barthelmess, C. A. Ellis, and J. Wainer. Workflow Modeling using Proclets. In *CoopIS00*, pages 198–209. Springer, 2000.

[17] W. M. P. van der Aalst, P. Barthelmess, C. A. Ellis, and J. Wainer. Proclets: A Framework for Lightweight Interacting Workflow Processes. *Int. J. Cooperative Inf. Syst.*, 10(4):443–481, 2001.

[18] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

 ISBN: 978-1-61208-109-0