Adaptive Workflows based on Flexible Assignment of Workflow Schemas and Workflow Instances – Extended Abstract –

Mathias Weske
Westfälische Wilhelms-Universität Münster
Steinfurter Straße 107, D-48149 Münster, Germany
weske@helios.uni-muenster.de

1 Introduction

Traditionally, workflow management deals with controlling the execution of application processes according to pre-defined specifications, known as workflow schemas [3, 8, 13]. This approach is well suited to support application processes with fairly static control structures. Real-world application processes, however, are not static in general. In contrast, they may require dynamic modifications to react quickly to new challenges imposed by the environment of the application process. While the exact definition of flexibility in workflow management systems is still under discussion [12, 7], it is widely accepted that dynamic modifications – or adaptations – of running workflows is an important feature of a flexible workflow management system [10, 2, 11]. This extended abstract sketches the conceptual design and implementation of dynamic modifications in the context of the WASA project at the University of Muenster.

2 The WASA$_2$ Approach

We use a workflow language based on process graphs, similar to those used by IBM's MQSeries Workflow (formerly IBM FlowMark). Workflows can be atomic or complex, there are data flow and control flow constraints between workflows, and technical and organizational information is ttached to workflow schemas. WASA$_2$ is based on an object-oriented approach: Workflow schemas and workflow instances are objects, which are characterized by a state and a behavior and which communicate with each other by sending and receiving messages. This general approach allows the flexible re-use of workflow schemas as sub-workflows in different complex workflow schemas, such that the embedding of the different occurrences of a workflow schema can be different with respect to its start condition and control flow and data flow constraints.

The object-oriented design and a distributed object middleware allow distributed workflow executions, such that workflow objects of a given workflow application can reside in different sites of a distributed computing system. In this case, workflows are controlled in a distributed manner without the need for a centralized workflow engine, which can become a performance bottleneck in large-scale workflow applications. In terms of flexibility, modeling workflow schemas and workflow instances as objects allows to change the association of a workflow instance with its controlling workflow schema. Hence, during its life time, a workflow instance can be controlled using different workflow schemas.

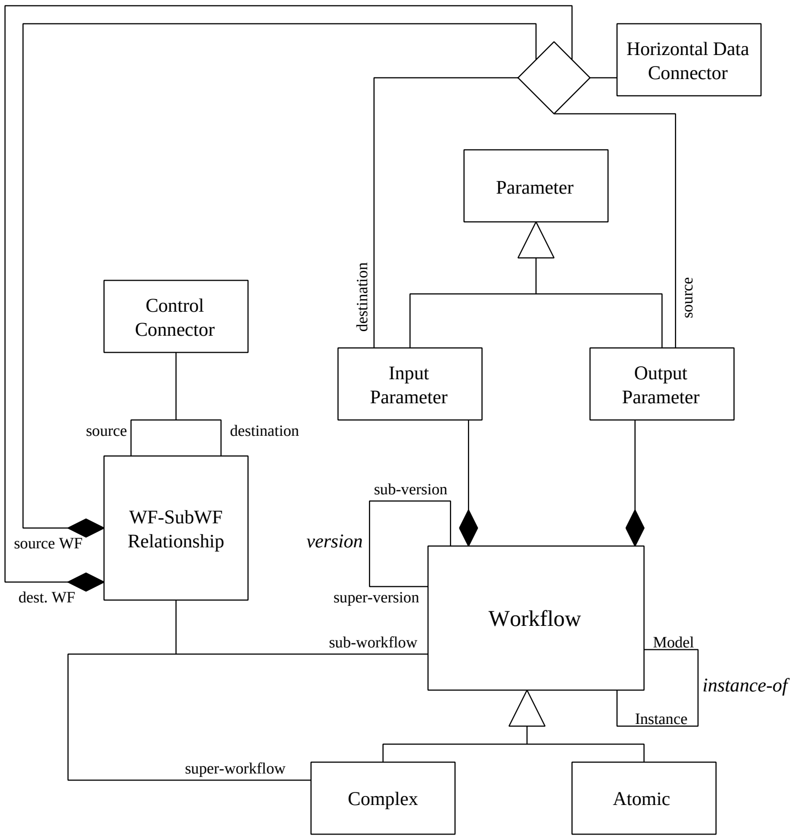


Figure 1: WASA$_2$ Workflow Meta Schema (Simplified Version).

The remainder of this section sketches the conceptual design of the system, which is specified in a workflow meta schema; a simplified version of the WASA$_2$ workflow meta schema is shown in Figure 1. Due to space limitations, we do not elaborate on different modeling alternatives for object-oriented workflow management systems [14], and we only discuss the parts of the workflow meta schema which are relevant for dynamic modifications.

The workflow class is in the center of the WASA$_2$ workflow meta schema; it contains workflow schema objects and workflow instance objects. Workflows can be either atomic or complex. The workflow hierarchy (i.e., the relationship between a complex workflow and its sub-workflows) is modeled by the WF-SubWF Relationship class, which defines a relationship between a complex workflow and a workflow, which can be complex or atomic. Workflow schemas and workflow instances are identified by states. The relationship between a workflow instance and the respective workflow schema is represented by an instance-of relationship. Each workflow schema can be associated with multiple workflow instances, while each workflow instance is associated with exactly one workflow schema at any given point in time. This relationship allows the flexible assignment of workflow instances to workflow schemas, as will be discussed in more detail in the remainder of this extended abstract.

Other parts of the meta schema deal with control connectors and data connectors; a control connector relates two WF-SubWF Relationship objects, defining execution order of the respective workflows. Each workflow has a set of input parameters and a set of output parameters, whose commonalities are represented in a Parameter class. Horizontal data flow represents data flow between workflows of a common super-workflow, while vertical data flow represents data flow between a super-workflow and its sub-workflows. The complete WASA$_2$ workflow meta schema as well as the design of the system and its implementation based on CorbaServices is presented in [15].

3 Dynamic Modifications in WASA$_2$

The ability to dynamically modify the structure of running workflow instances is an important feature of a flexible workflow management system, since it allows running workflow instances to adapt to changes in the environment. In this context, "environment" refers to the market environment of processes, including new services provided by competitors, new and faster or more cost efficient ways to produce or deliver goods, and providing new services or parts thereof. Besides changes in the market, there may be new legal regulations that have to be implemented by application processes. For instance, consider a new legal regulation that a single checking mechanism has to be changed to a double-checking mechanism. As a consequence, workflow instances have to use the new checking policy in order comply with the new regulations. Changes in the technical environment of the process are another motivation for dynamic modifications. Assume there are new tools available to perform tasks more efficiently then the active and all future workflow instances should make use of the new infrastructure.

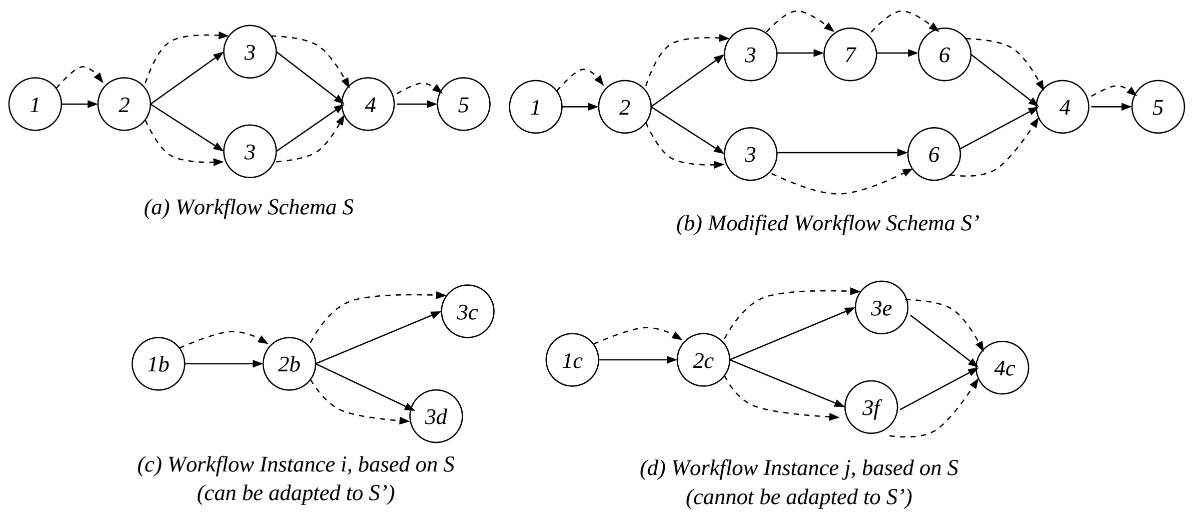
In dynamic modifications, it is important to define correctness criteria which determine if and when a workflow instance can be adapted to a new workflow schema. To motivate our correctness criterion, we start by discussing correctness in workflow applications in general, i.e., without dynamic modifications. Since in the workflow context, generic correctness properties like in database transaction processing (e.g., conflict serializability, recoverability) do not suffice to describe correct workflows, application specific correctness criteria have to be defined. From an application-oriented point of view, these criteria are specified in business process models, which describe which activities have to be performed, and what are the constraints between them. When supporting business processes by workflow technology, the correctness of workflow instances is specified in workflow schemas. Hence, a workflow execution is correct if and only if it satisfies the criteria specified in the respective workflow schema. Control flow and data flow constraints as well as role information and technical information are examples of properties which are specified in workflow schemas and which have to be met by workflow instances. The task of a workflow management system is to make sure each workflow instance is executed according to its workflow schema.

Based on this perception of correctness in the workflow context, the approach to controlling dynamic changes in WASA$_2$ is fairly simple, yet effective:

> *A workflow instance can be dynamically modified, i.e., it can be adapted to a new workflow schema, if the workflow instance could have been controlled from the beginning using the new workflow schema.*

For an example consider Figure 2, which shows a workflow schema S (a), a modified workflow schema S' (b) and two currently active workflow instances i and j, based on the original workflow schema S ((c) and (d), resp.). Notice that workflow instances i and j are correct with respect to workflow schema S, since they can be continued according to S. We now assume that there is a dynamic modification, changing workflow schema S to S', shown in Figure 2(b).

When a workflow administrator decides to change a workflow dynamically, he or she first suspends the execution of the workflow. This is necessary, since otherwise race conditions between the normal



(a) Workflow Schema S

(b) Modified Workflow Schema S'

(c) Workflow Instance i, based on S (can be adapted to S')

(d) Workflow Instance j, based on S (cannot be adapted to S')

Figure 2: Workflow Schemas S and S' and Workflow Instances i, j, based on S.

workflow execution and the correctness checks would occur. In our example, given workflow instances i and j (so far based on S), the system now has to decide whether these can be continued with the modified workflow schema S'. This check is performed by analyzing the current state of the workflow instances and matching the structure of the new workflow schema S' against the states.

We first consider workflow instance i. It is obvious that the workflow instance can be adapted to S', since there is a mapping between the sub-workflow instance already executed on behalf of i and sub-workflow schemas of S'. More technically, the decision whether or not a workflow instance can be adapted to a new workflow schema is taken based on a mapping. The sub-workflow instances which have already been executed are mapped to the sub-workflow schemas of the new workflow schema. In our example, sub-workflow instance $1b$ of workflow instance i is mapped to sub-workflow schema 1 of workflow schema S', and sub-workflow instance $2b$ is mapped to sub-workflow schema 2 of S'. The two workflow schemas based on 3 can be mapped to $3a$ and $3b$. Since furthermore the control flow constraints of the workflow instance and the workflow schema comply, and assuming that data flow constraints also comply, a mapping can be found. As a consequence, i can be continued to become a complete workflow instance based on S': after the termination of $3c$, an instance $7a$ (based on workflow schema 7) and an instance $6a$ (based on workflow schema 6) can be started, and after the completion of $3d$ an instance $6b$ (based on workflow schema 6) can be executed. Finally, workflow instances based on workflow schemas 4 and 5 can be performed sequentially. Hence, the resulting workflow instance is correct and complete with respect to the new workflow schema S'.

Along the lines of this argumentation it is clear that j cannot be adapted to the new workflow schema. It proceeded further than i; in particular, it already started a sub-workflow instance based on workflow schema 4, i.e., $4c$. Since in the modified workflow schema S', 4 can only be stated after additional sub-workflow instances have completed, workflow instance j (d) violates this requirement. Even starting sub-workflow instances for 7 and 6 (as specified in S') right away would not help, since by the control flow constraints specified in S', an instance of 4 cannot start until workflow instances based on workflow schemas 7 and 6 have been executed. This constraint imposed by workflow schema S' cannot be satisfied by any continuation of j. Hence, j cannot be modified dynamically with respect to workflow schema S'.

In WASA$_2$, the instance-of relationship associates a workflow instance object with a workflow schema object. At each point in time, each workflow instance object is associated with exactly one workflow schema object, while each workflow schema object can be associated with an arbitrary number of workflow instance objects. Given this organization, dynamic modifications can be implemented by changing the respective instance-of relationship objects at runtime. To implement a dynamic modification based on a workflow schema S with numerous currently active workflow instances, the following steps are carried out:

- create a new workflow schema (or use an existing one) S'
- based on the instance-of relationship, let C be the set of all workflow instances which are associated with S
- compute a set $C' \subseteq C$ of workflow instances which can be adapted to the new workflow schema S', based on finding a mapping as sketched above
- allow a workflow administrator to select a subset of C' of workflow instances, which will actually be dynamically modified; update instance-of relationship of these workflow instances accordingly
- all workflow instances are continued using their respective instance-of relationships, i.e., modified workflow instances are continued with S', and non-modified workflow instances are continued with the original workflow schema S

To perform an adaptation of workflow instance i to S', sub-workflow instances which are no longer needed are deleted, and new sub-workflow instance objects are created, as specified in the new workflow schema S'. These workflow instance objects are embedded in the context of the complex workflow instance by creating the respective WF-SubWF Relationship objects. In our example, new workflow instance objects $7a$, $6a$, $6b$, $4d$ and $5b$ are created and attached to the complex workflow using WF-SubWF Relationship objects. The workflow is continued with the execution of the sub-workflow instances $7a$ and $6a$.

4 Related Work

Two recent workshops were devoted to adaptive and flexible workflow management [7, 12]. In [5] a taxonomy of adaptive workflow management is proposed. In particular, the constantly changing market environment of business processes is regarded as a major motivation for flexible workflow management. Process level adaptations and resource level adaptations are among the requirements for a flexible workflow management system. Techniques for exception handling in workflow management systems are identified and classified in [9, 1]. An approach to enhance the flexibility of workflow management systems based on an integration of workflow and workspace management techniques is discussed in [6]. To classify flexibility requirements, *a priori* and *a posteriori* flexibility is characterized by properties of the application that are known before it starts and after it has started, resp.

In [10] a workflow language ADEPT is proposed, which allows to specify workflow schema using symmetric graphs. There are different node types, reflecting for instance split and join nodes, and start and end nodes of loops. Based on this workflow language, a model ADEPT$_{flex}$ supporting a set of operations to change the structure of workflows is defined, allowing to dynamically change workflows, while keeping their symmetric structure. ADEPT does not consider workflow schemas; only workflow instances are discussed. Hence, the approach does not consider dynamic changes in presence of multiple workflow instances based on a modified workflow schema. There is an operational running prototype implementing dynamic modifications according to ADEPT$_{flex}$. In [4], a Petri-Net based approach to model workflows which includes flexibility mechanisms is proposed. Simple dynamic modifications are allows, for instance to leave unspecified defined portions of the net to be filled when the workflow executes; this is denoted by late modeling; this functionality is implemented in the CORMAN prototype [4].

5 Conclusions

This extended abstract sketches the design of controlled dynamic modifications of workflow instances by a flexible assignment of workflow schemas to workflow instances. By allowing to change the assignment of workflow instances and workflow schemas at different points in time, different schemas can be used to control a workflow. An adaption of a workflow instance i to a workflow schema $\mathcal{S}$ can be done whenever i can be continued such that it fits $\mathcal{S}$. This elegant yet simple characterization of the correctness of a dynamic modification allows to maintain the correctness property of workflow instances with respect to workflow schemas, while providing workflow flexibility by dynamic modifications. Future work in the WASA project will be centered around user interface design, and we plan to use the WASA$_2$ system in real-world workflow applications which make use of the capabilities of the system.

References

[1] Deiters, W., Goesmann, T., Just-Hahn. K., Lffeler, T. Rolles, R.: *Support for exception handling through workflow management systems.* In Proceedings CSCW-98 Workshop: Towards Adaptive Workflow Systems. (downloaded from http://ccs.mit.edu/klein/cscw98/paper19 on 09-24-1998)

[2] Ellis, C., K. Keddara, G. Rozenberg: *Dynamic Change Within Workflow Systems.* In Proc. Conference on Organizational Computing Systems (COOCS) 1995, 10–22

[3] Georgakopoulos, D., M. Hornick, A. Sheth: *An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure*. Distributed and Parallel Databases, 3:119–153, 1995

[4] Hagemeyer, J., T. Herrmann, K. Just-Hahn, R. Striemer: *Flexibility in Workflow Management Systems (in German).* Software-Ergonomie '97, 179–190, Dresden, March 1997.

[5] Han, Y., Sheth, A., Bussler, C.: *A Taxonomy of Adaptive Workflow Management.* In Proceedings CSCW-98 Workshop: Towards Adaptive Workflow Systems. (downloaded from http://ccs.mit.edu/klein/cscw98/paper03 on 09-24-1998)

[6] Joeris, G.: *Aspects and Concepts of Flexibility in Workflow Management Systems.* (in German) In Proc. D-CSCW98 Workshop on Flexibility and Cooperation in Workflow Management Systems, Dortmund, Sept 1998. Technical Report Angewandte Mathematik und Informatik 18/98-I, University of Muenster, Germany 1998

[7] Klein, M. (Ed.): *Towards Adaptive Workflow Systems.* Workshop in The 1998 ACM Conference on Computer Supported Cooperative Work, Seattle, Washington, November 14-18, 1998 (Online Proceedings at http://ccs.mit.edu/klein/cscw98/ on 09-24-1998)

[8] Leymann, F., W. Altenhuber: *Managing Business Processes as an Information Resource*. IBM Systems Journal 33, 1994, 326–347

[9] Luo, Z., Sheth, A.: *Defeasible Workflow, its Computation and Exception Handling.* In Proceedings CSCW-98 Workshop: Towards Adaptive Workflow Systems. (downloaded from http://ccs.mit.edu/klein/cscw98/paper10 on 09-24-1998)

[10] Reichert, M., P. Dadam: *Supporting Dynamic Changes of Workflows Without Loosing Control.* Journal of Intelligent Information Systems, Special Issue on Workflow and Process Management, Vol. 10, No. 2, 1998

[11] Sheth, A., D. Georgakopoulos, S.M.M. Joosten, M. Rusinkiewicz, W. Scacchi, J. Wileden, A. Wolf: *Report from the NSF Workshop on Workflow and Process Automation in Information Systems*. Technical Report UGA-CS-TR-96-003 University of Georgia, Athens, GA, 1996

[12] Siebert, R., Weske, M. (Eds.): *Flexibility and Cooperation in Workflow Management Systems.* (in German) Workshop at D-CSCW98, German Conference on CSCW 1998, Dortmund, Sept 1998. Technical Report Angewandte Mathematik und Informatik 18/98-I, University of Muenster, Germany 1998

[13] Weske, M., Vossen, G.: *Workflow Languages*. In: P. Bernus, K. Mertins, G. Schmidt (Editors): Handbook on Architectures of Information Systems. (International Handbooks on Information Systems), pp 359–379. Berlin: Springer 1998

[14] Weske, M., Hündling, J., Kuropka, D., Schuschel, H.: *Object-Oriented Design of a Flexible Workflow Management System*. (in German) Informatik Forschung und Entwicklung. 13(4) 1998, pp 179–195. Berlin: Springer 1998

[15] Weske, M.: *Design and Implementation of an Object-Oriented Workflow Management System.* Fachberichte Angewandte Mathematik und Informatik 33/98-I, Universität Münster 1998