Formal Foundation and Conceptual Design of Dynamic Adaptations in a Workflow Management System

Mathias Weske
Westfälische Wilhelms-Universität Münster
Steinfurter Straße 109, D-48149 Münster, Germany
weske@helios.uni-muenster.de

Abstract

While the different aspects of flexible workflow management are still under discussion, the ability to adapt the structure of running workflow instances to modified workflow schemas is an important property of a flexible workflow management system. In this paper we present the formal foundation and conceptual design of dynamic adaptations in an object-oriented workflow management system. We describe in some detail how workflow schemas are represented. The system architecture based on the CORBA object-oriented middleware is overviewed, and the implementation of dynamic adaptations is sketched. An example introduces the graphical user interface of the system and shows a dynamic adaptation.

1 Introduction

Traditionally, workflow management aims at modeling and controlling the execution of application processes according to predefined specifications, known as workflow schemas [3, 7, 10]. This approach is well suited to support application processes with static control structures, which are not expected to change frequently during workflow enactment. However, given the dynamic structure of today's organizations in commerce, public administration, and in science and engineering, it is unlikely that application processes are modeled once to be executed repeatedly without any changes. Reasons for changes include new challenges imposed by the technical, organizational, or by the market environment of application processes. While flexibility is still under discussion in the workflow community (e.g., [17, 8]), it appears that dynamic adaptations of running workflows to new workflow specifications is an important feature of a flexible workflow management system, as discussed, e.g., in [2, 13, 1, 16]. In this paper we present the formal foundation and conceptual design of dynamic adaptations in an object-oriented workflow management system. The system architecture based on an object-oriented middleware is overviewed, and the implementation of dynamic adaptations is sketched.

This paper is organized as follows. Section 2 introduces a workflow language based on a variant of directed graphs. The conceptual design of dynamic adaptations is presented in Section 3, followed by a brief discussion of implementation aspects. A section on related work and a discussion of future perspectives complete this paper.

2 Foundation: Workflow Schemas and Workflow Instances

Formalizations of the workflow-relevant aspects of application processes are known as workflow schemas, which are expressed in workflow languages [18]. This section introduces a workflow language based on a variant of directed graphs. The workflow language is similar to the one used in MQSeries Workflow [5], IBM's workflow management system; the relationship between these workflow languages and systems are discussed in Section 5. In addition to workflow schemas, workflow instances as representations of workflow-relevant components of running application processes are formalized, since the correctness of workflow instances in the presence of dynamic adaptations is characterized with respect to workflow schemas.

Workflow schemas are based on directed graphs, whose nodes represent workflow schemas and whose edges represent constraints between them, namely data flow and control flow. Workflow schemas can be atomic or complex; while atomic workflows do not have an internal structure, complex workflows consist of a set of workflow schemas, each of which can be atomic or complex, resulting in a hierarchical structure called a workflow schema tree: The root of a workflow schema tree represents the top-level workflow, the inner nodes represent other complex workflows, and the leaves a the workflow schema tree represent atomic workflows. The relative position of workflows is characterized by the terms sub-workflow and super-workflow, respectively.

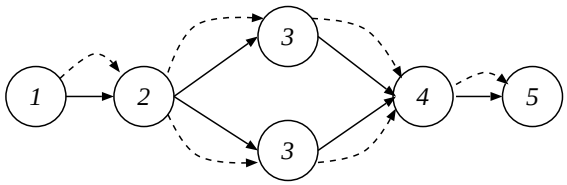


Figure 1. Workflow Schema Graph.

Data dependencies between workflows, i.e., the transfer of information between workflows is specified as data flow. In the workflow language presented, data flow is based on typed parameters. Each workflow schema is assigned a set of input parameters and a set of output parameters. When starting a workflow, its input parameters are read. On the termination of a workflow, its results are written into its output parameters.

There are two constructs to define workflow behavior: Execution order constraints and start conditions. Sibling workflows can be related to each other by execution order constraints, expressed by control connectors. This form of execution constraint is also known as control flow. The conditions under which a given workflow instance is executed is defined by its start condition. By evaluating the start condition of a workflow, the system decides whether it has to be executed during a particular application process. Clearly, a true value of the start condition makes the workflow execute, while the workflow is skipped if the start condition evaluates to false. To specify start conditions, workflow execution data as well as application-specific data can be used.

Figure 1 shows a complex workflow schema graph including data connectors (dotted arcs) and control connectors (solid arcs). For clarity, parameters are omitted. The execution constraints specify that sub-workflow 1 executes first, followed by 2 and a parallel execution of two workflows based on workflow schema 3, etc. Notice that in this paper we restrict our attention to one level of abstraction, i.e., one level of the workflow hierarchy is discussed. This allows to concentrate on horizontal data flow (data flow between workflows in one level), while omitting issues related to vertical data flow (data flow between a super-workflow and its sub-workflows. We remark that the results presented in this paper also apply to hierarchically structured workflows, as is described in detail in [20].

A formal specification of workflow schemas and workflow instances is introduced (due to space limitations, vertical data flow and some technical details are omitted [20]). A workflow schema node s consists of a set in_s of typed input parameters, a set out_s of typed output parameters, a list sip_s of typed start input parameters, and a boolean function $sc_s : sip_s \mapsto \{true, false\}$, the start condition.

Definition 1 A *workflow schema graph* G_s is defined by $G_s = (V_s, C_s, D_s)$, such that

- V_s is a set of workflow schema nodes, $s \in V_s$
- C_s is a set of control connectors between parameters of sibling workflows, $C_s \subseteq out_j \times sip_k$. The fact that $(p, q) \in C_s$ and $p \in out_j$, $q \in sip_k$ is graphically represented by a directed edge with source node j and destination node k.
- D_s is a set of (horizontal) data connectors $D_s \subseteq out_j \times (in_k \cup sip_k)$. The fact that $(p, q) \in D_s$, such that p is a parameter of j, and q is a parameter of k is graphically represented by a directed dotted edge with source node j and destination node k.

$\diamond$

To motivate consistency criteria on the syntactic structure of workflow schemas, consider a complex workflow schema with sub-workflow schemas i and j, a control connector $i \rightarrow j$, and a data flow between these workflow schemas in the opposite direction. A workflow schema with these properties leads to a deadlock-like situation, in which two workflow activities mutually wait for each other to terminate: j waits for i due to control flow, and i waits for j due to data flow. To avoid situations like this one, a set of consistency criteria are defined. A workflow schema graph G_s is consistent, if the following criteria are satisfied:

1. *Completeness:* For each input parameter $q \in in_j$ there is a data flow connector $(p, q) \in D_s$, providing input data for j.
2. *Type compatibility:* Data flow connectors link parameters of the same data type.
3. *Data availability:* For each horizontal data connector connecting an output parameter of i and an input parameter of j there is a path of control connectors from i to j. We say that data flow follows control flow.
4. *Acyclic structure:* Control connectors of a given workflow schema graph do not form cycles. Since data flow follows control flow, the structure of data flow constraints is also acyclic.

Analogously to workflow schemas, workflow instances are specified by defining workflow instance nodes and workflow instance graphs. If a workflow instance i is based on a workflow schema s then SchemaOf$(i) = s$. A workflow instance node i based on a workflow schema node s consists of a parameter value $p \in Dom(t)$ for each parameter of type t in sip_s, in_s, and out_s, respectively, and a start condition $sc_s(sip_i) \in \{\text{true, false}, \perp\}$, based on the values of the start input parameters; $\perp$ represents the undefined value, which a parameter takes before it is provided a value. A start condition has the undefined value before it is evaluated.

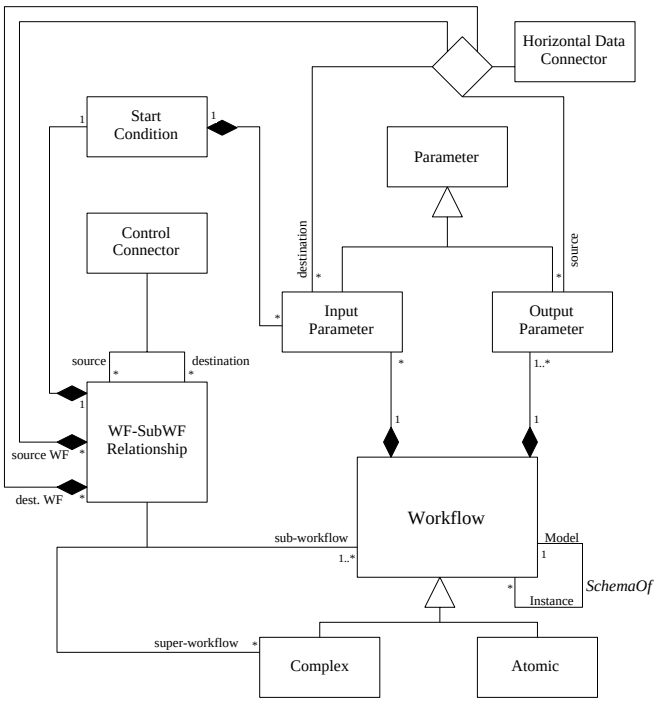


Figure 2. Workflow Meta Schema (Simplified Version).

Definition 2 A *workflow instance graph* $G_i = (V_i, C_i, D_i)$ based on a workflow schema graph $G(s)$ is defined as follows:

- For each workflow schema node $s' \in V_s$ there is a workflow instance node $i' \in V_i$, such that SchemaOf$(i') = s'$.
- For each control connector in C_s there is a control connector in C_i, linking the respective parameters.
- For each data connector in D_s there is a data connector in D_i, linking the respective parameters.

$\diamond$

3 Conceptual Design

Based on the formal foundation presented, this section introduces the conceptual design of a flexible workflow management system using an object-oriented approach. Based on a class diagram, the conceptual design of dynamic adaptations is developed, and correctness criteria for dynamically adapted workflow instances are presented.

3.1 Workflow Meta Schema

In the object-oriented approach to modeling complex software systems, entities of the application domain are represented by objects and grouped in classes, encapsulating their structure and behavior. Since the domain of interest is workflow management, workflow schema objects, workflow instance objects, and objects describing the organizational and technical execution environment of workflows are in the center of attention.

It is important to stress that workflow schemas and workflow instances are both modeled as first-class objects, which are characterized by a state and a behavior and which communicate with each other by sending and receiving messages. This general approach, to our knowledge, is unique in workflow management — it 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 start condition, and control flow and data flow constraints. In addition, changes to workflow schemas can be implemented by value changes, which in general are much easier to handle than changes on the schema level. A flexible assignment of workflow schemas and workflow instances allows to adapt workflow instances to new workflow schemas, which is our main focus in this paper.

The conceptual design of the system is described by a class diagram, which models workflow instances and workflow schemas. Since it represents the structure of workflow schemas, it can be regarded as a workflow meta schema. A simplified version of the workflow meta schema is shown in Figure 2. This paper does not elaborate on different modeling alternatives for object-oriented workflow management systems, and only the parts of the workflow meta schema are discussed which are relevant to dynamic adaptations; the complete workflow meta schema is discussed in [19]. The Unified Modeling Language (UML) [12] is used for the object-oriented modeling and design. The workflow class is in the center of the workflow meta schema. As described above, it contains workflow objects which are either workflow schema objects or workflow instance objects. The workflow hierarchy is modeled by the WF-SubWF Relationship class, which defines a set of relationships between complex workflows and workflows, each of which can be complex or atomic. Workflow schemas and workflow instances are identified by different states of workflow objects. The relationship between a workflow instance and the respective workflow schema is represented by the SchemaOf relationship, a recursive relationship in the workflow class. This relationship allows the flexible assignment of workflow instances and workflow schemas, as required by dynamic adaptations of workflow instances.

We remark that the workflow meta schema models all artifacts of the formal specification of workflow schemas and

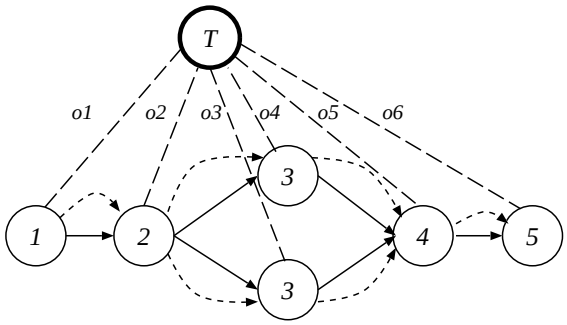


Figure 3. Representation of Workflow Schema in Workflow Meta Schema.

workflow instances as presented in Section 2. Formal correctness criteria of dynamic adaptations can be specified in the formal notation, while the development of a complex software system can make use of object-oriented design techniques.

In order to illustrate the representation of workflow schemas and workflow instances in the workflow meta schema, consider Figure 3, where a complex workflow schema T is shown. (To keep the presentation clear, parameters and start conditions are omitted. However, they are represented in the workflow meta schema.) In general, the sub-workflow schemas of a complex workflow schema are represented by objects of the WF-SubWF-Relationship class. In Figure 3, these objects are represented by dotted lines marked with $o1$ through $o6$. Notice that workflow schema 3 occurs twice as a sub-workflow schema of T; these occurrences are represented by the two WF-SubWF-Relationship objects $o3$ and $o4$. Turning to workflow instances, for each occurrence of a sub-workflow schema in T, a workflow instance based on T contains a sub-workflow instance. An object of the SchemaOf class links the workflow schema object to the workflow instance object.

3.2 Dynamic Adaptations

As introduced above, workflow schemas can be regarded as formal representations of the automated parts of application processes. A workflow instance is therefore correct if it satisfies the constraints imposed by the respective workflow schema. As a consequence, a workflow management system has to guarantee that each workflow instance satisfies the criteria defined by the respective workflow schema. This general notion of correctness in workflow systems is adequate, since the correctness is defined by the domain experts in business process models and — in the workflow side — in workflow schemas.

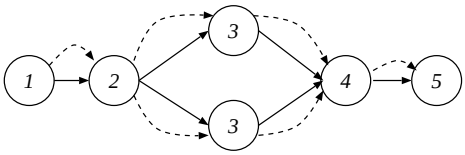
The definitions of workflow schemas and workflow instances as given above are valuable for correctness in the presence of dynamic adaptations, since they govern whether a given workflow instance is correct with respect to a given workflow schema. As will be explained below, the mathematical formalization serves as a basis for deciding whether a dynamic adaptation is correct, i.e., whether a given workflow instance can be adapted to a new workflow schema. It can easily be seen that the mathematical formalization and the workflow meta schema, specified as a class diagram comply. For instance, complex workflow schemas consist of workflow schemas, which can be atomic or complex; there are control flow and data flow constraints between workflow schemas in the context of a given complex workflow schema.

Before formalizing the concepts, an example of a dynamic adaptation shows the rational behind these considerations. Consider workflow schema s shown in Figure 4(a) and workflow schema s' (b), which contains additional sub-workflow schemas. In Figure 4(c), a workflow instance based on workflow schema s is shown. A shading is used to indicate sub-workflow instances which have already started. Hence, all sub-workflow instances except for the last sub-workflow instance of i have started. Due to the control flow constraints defined in s, at least workflow instances $1a$, $2a$, $3a$ and $3b$ have already terminated. A criterion for deciding whether a complex workflow can be adapted to a workflow schema can be defined as follows:

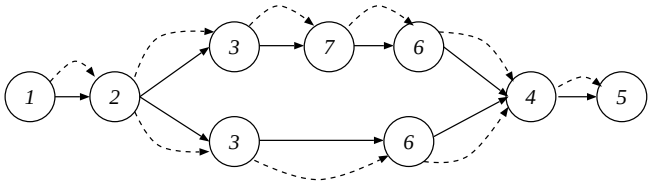
Definition 3 A complex workflow instance i *is adaptable to* a workflow schema s' if and only if there is a continuation of i such that i complies with s'. ◇

This rather informal definition of workflow instance adaptability is used to decide whether i can be adapted to s'. In the workflow instance shown, the adaptation is not possible, since sub-workflow instance $4a$ has already started. To comply with s', additional sub-workflow instances have to be executed *before* $4a$ can start, namely a sub-workflow instance for 7 and two workflow instances based on workflow schema 6. Hence, there is no continuation for the complex workflow instance that satisfies the control flow constraints imposed by the new workflow schema. Consequently, i is not adaptable to s'. Workflow instance j shown in Figure 4(d) is adaptable to s', since there is a continuation of j that is consistent with s'. After the check is performed, new sub-workflow instances can be created, as well as their embedding in the context of the complex workflow. The resulting workflow instance is shown in Figure 4(e).

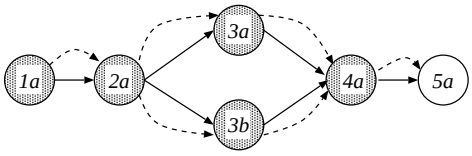
To put these considerations more formally, the decision whether or not a workflow instance can be adapted to a new workflow schema is based on mappings. In particular, the sub-workflow instances which have already been started are mapped to the sub-workflow schemas of the new complex workflow schema. In addition, control flow and data flow constraints also have to be taken into account. In our exam-



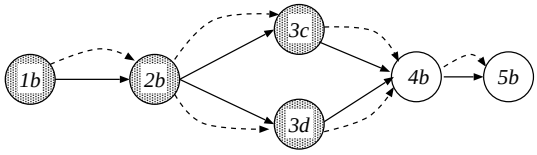
(a) Workflow Schema s



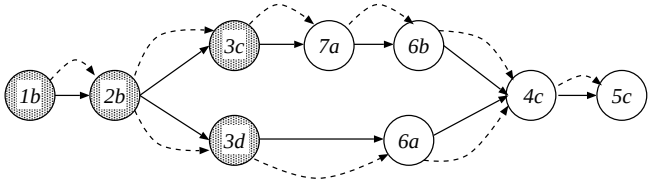
(b) Workflow Schema s'



(c) Workflow Instance i



(d) Workflow Instance j



(e) Workflow Instance j, Adapted to s'

Figure 4. Workflow Schemas (a), (b), Workflow Instances (c), (d) and Dynamically Adapted Workflow Instance (e).

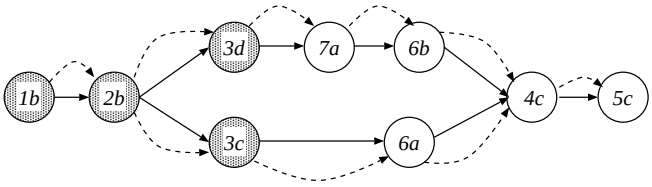


Figure 5. Alternative Mapping in Dynamic Adaptation.

ple, sub-workflow instance $1b$ of j is mapped to sub-workflow schema 1 of s', and sub-workflow instance $2b$ is mapped to sub-workflow schema 2 of s'. The two sub-workflow instances $3c$ and $3d$ are mapped to the occurrences of workflow schema 3 in s'. Since furthermore the control flow and data flow constraints of the workflow instance and the workflow schema comply, a mapping exists, and j is adaptable to s'.

In fact, there are two mappings with this property, due to the fact that the sub-workflow instances $3c$ and $3d$ can be mapped arbitrarily to the two occurrences of sub-workflow schema 3 in s'. In the first mapping alternative, workflow instance $7a$ is executed after $3c$, and workflow instance $6a$ is executed after $3d$. In the second alternative, $7a$ is executed after $3d$, while $6a$ is executed immediately after $3c$. While the workflow management system can compute the set of possible mappings for a given dynamic adaptation, the decision on which mapping to choose has to be taken by the person responsible for the dynamic adaptation, based on application specific knowledge and the aim of the dynamic adaptation. The workflow instance resulting from the second mapping alternative is depicted in Figure 5.

3.3 A Formal Correctness Criterion

After introducing the general idea of correct dynamic adaptations based on mappings, a more detailed view on the subject is taken. We observe that to check for adaptability of a workflow instance, only the sub-workflow instances which have already been started have to be taken into account. This means that workflow instance which have not yet been started, i.e., for which StateOf(i) =notStarted holds, can be purged from the workflow instance graph. Given a workflow instance i and a set $O_i \subseteq V_i$ of obsolete workflow instance nodes of i, such that $O_i := \{j \mid j \in V_i \wedge \text{StateOf}(j) = \text{notStarted}\}$ the purged workflow instance graph of i can be constructed as follows:

- purge from V_i all obsolete workflow instance nodes: $V_i := V_i - O_i$,
- purge all control flow connectors and data flow connectors from C_i and D_i with a source or target node $j \in O_i$.

Definition 4 Let i be a workflow instance and s a workflow schema. A function $m : V_i \mapsto V_s$ is *a mapping from i to s* if for each workflow instance node $j' \in V_i$ there is a workflow schema node $j \in V_s$, such that $m(j') = j \Rightarrow \text{SchemaOf}(j') = j$ and each workflow instance node is mapped to one and only one workflow schema node, i.e., $m(j') = m(k') \Rightarrow j' = k'$ for $j', k' \in V_i$. ◇

We remark that for a workflow instance i, whose execution is controlled by a workflow schema s, a mapping from i to s exists. This observation is based on the fact that a complex workflow instance based on a given complex workflow schema is initiated by creating for each sub-workflow schema of the complex workflow schema a sub-workflow instance, and control flow and data flow constraints according to the workflow schema.

The argumentation continues as follows: Based on a mapping from the purged workflow instance to the new workflow schema, valid mappings are defined. A mapping is valid if the execution constraints found in the workflow schema are matched by the workflow instance, and vice versa. This requires to take into account execution order constraints specified by control connectors as well as data constraints defined by data connectors. In particular, for each control connector in the workflow schema s' connecting workflow schema nodes which have corresponding workflow instance nodes in i, there has to be a corresponding control connector in the workflow instance. If there is no such control connector then the respective workflow instances are either executed concurrently or there is an additional workflow instance which is performed in between them. This situation has to be ruled out for valid mappings. Regarding properties of valid mappings with respect to data flow, for each data connector in the workflow instance there has to be a corresponding data connector in the workflow schema

Definition 5 Given a purged workflow instance i, a mapping m from i to s' is a *valid mapping*, if the following conditions hold:

- For each control connector in s' connecting two sub-workflow schemas which have corresponding sub-workflow instances in V_i, there is a control connector in C_i, and vice versa.
- For each data connector in workflow instance i, there is a data connector in workflow schema s'. Analogously, for each data connector in the workflow schema connecting two sub-workflow schemas which have corresponding sub-workflow instances in V_i, there is a corresponding data connector in the workflow instance

◇

Based on the definition of a valid mapping, it is now obvious how adaptability of workflow instances is characterized: by the presence of a valid mapping between a workflow instance and a new workflow schema.

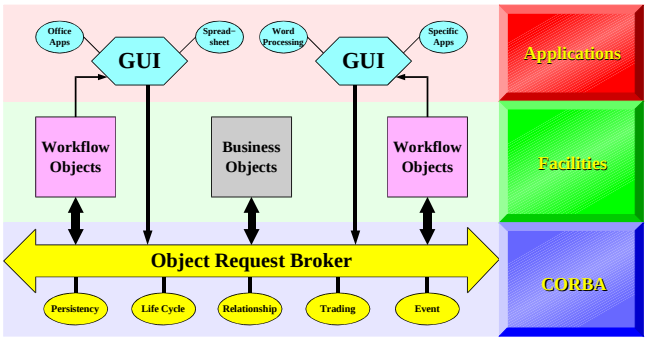


Figure 6. System Architecture.

Theorem 1 A workflow instance i can be adapted to a workflow schema s' if there is a valid mapping m of i with respect to s'.

This theorem is proven by using workflow instance i and the valid mapping of i with respect to s' to transform i such that it complies with workflow schema s'. Details of the proof can be found in [20].

4 Implementation Aspects and Example

Since the conceptual design is based on object-oriented principles, the system design rather closely resembles the conceptual design. In fact, the classes shown in the workflow meta schema are implemented in the system. For instance, there is a generic workflow class, which holds workflow schema objects and workflow instance objects. The composition of complex workflows as a set of workflows with control flow and data flow constraints is specified by the WF-SubWF Relationship class and classes related to it, for instance the Control Connector class.

The system is implemented in Java JDK using the OrbixWeb [6] CORBA object request broker implementation by Iona Technologies. An overview of the system architecture is shown in Figure 6. It is a layered architecture, composed of three levels: The applications, the facilities, and the foundation. In the application layer, a workflow client application resides, through which workflow participants and workflow administrators access the system. External applications are also in this level, although they are not considered part of the system. Examples for external application programs are office applications like text processing or spreadsheet applications.

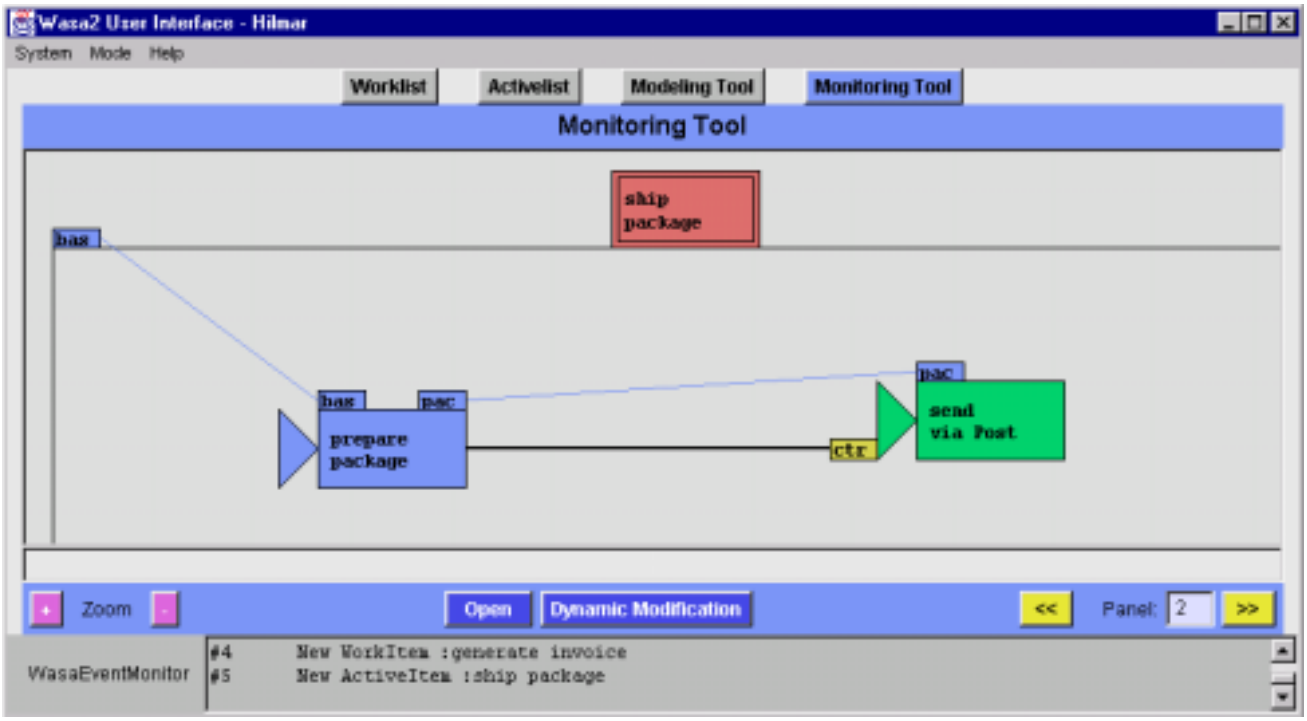


Figure 7. Workflow Instance to Change Dynamically.

The facilities level is the core layer of the architecture, where workflow objects and related objects reside. For instance, a WF-SubWF Relationship class holds objects which represent the structure of complex workflows, a class for horizontal data flow represents the relationships between parameter objects. Classes for agents and roles hold information on the organizational context. Data type and parameter instance classes keep track of the information aspect. The basic objects and services to implement the higher-level objects are implemented in the foundation layer. This layer consists of CORBA Services, which we have implemented. Details on the implementation of a persistent storage service can be found in [21]; individual implementation aspects appear in [14, 4, 9, 15].

Dynamic adaptations are implemented by a flexible assignment of workflow schemas and workflow instances using specific SchemaOf relationships. The steps that are involved in a dynamic adaptation of a workflow instance i to a workflow schema s' are given by:

1. *Suspend:* To suspend i, all sub-workflow instances of i which are in the running state are completed, but no additional sub-workflow instances are started.

2. *Check:* Using valid mappings, the system checks whether a dynamic adaptation of workflow instance i to workflow schema s' is possible. In case multiple adaptations are possible, the person responsible for the dynamic adaptation selects one alternative.

3. *Adapt:* This step involves the deletion of obsolete sub-workflow instances and, typically, the creation of new sub-workflow instances with accompanying data flow and control flow connectors, as specified in s'.

To illustrate dynamic adaptations in an example and to sketch the graphical user interface of the system, a sample workflow from electronic commerce is presented. The workflow is called E-Bookshop, and it describes the activities to process an online book order in an electronic bookshop. Using natural language, the toplevel workflow is described as follows:

Users access the system via standard web browsers to retrieve information and to order books. The workflow starts with a register order sub-workflow, which retrieves the ordering information from a Java applet which is interpreted by a web browser. At this point a branching occurs: Either a checking workflow is performed (if the data seem corrupted) or a check store workflow is performed immediately after the register order workflow to check whether the books ordered are available. After the store is checked and assuming the books are available, two sub-workflows are executed concurrently: a generate invoice workflow and a ship package workflow.

The graphical user interface [4, 14] shown in Figure 7 displays the internal structure of the ship package workflow

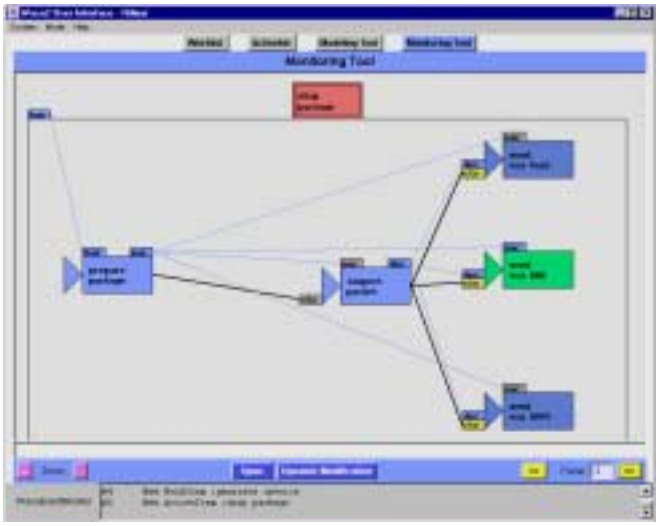


Figure 8. Workflow Instance is Continued Using Dynamically Modified Workflow Schema.

instance, which we will dynamically adapt shortly. In the original workflow schema, there are just two sub-workflows which are executed sequentially, as shown in Figure 7. Now new competitors emerge in the postal service market, and the bookstore wants to make use of these new services for the current workflow instance and all future ones. Since the ship package sub-workflow is already completed, the current workflow instance has to be adapted dynamically in order to make use of the new postal services. At this point in time, the send via post workflow instance appears in the work list of one or more agents. To check if the dynamic adaptation can be permitted, this workflow instance has to be suspended. The work items of suspended workflow instances are deleted from all work lists.

In the new version of the sub-workflow schema, an inspect package workflow is added, followed by three alternatives for sending the parcel, using different service providers. To decide whether a dynamic adaptation is feasible, the system checks whether the workflow instance under consideration can be continued to become a complete and correct workflow instance with respect to the new schema, i.e., the system tries to find a valid mapping of the workflow instance to the new workflow schema. As is shown in Figure 7, the workflow instance has already executed its first sub-workflow "prepare package". At this point a valid mapping between the workflow instance and the new workflow schema is computed. In this case, exactly one valid mapping can be found, since the only sub-workflow instance completed is the prepare package workflow, which is also present in the new workflow schema, and all added sub-workflows will be executed after it.

To perform the dynamic adaptation, the SchemaOf relationship of the workflow instance is changed to the new workflow schema, and the additional sub-workflow instances are created. From a technical point of view, this adaptation requires a variety of additional activities. Besides creating new sub-workflow instances and deleting superfluous ones, the control flow and data flow objects which are no longer necessary are deleted, and new control flow and data flow objects are created. When these activities are completed, the dynamically adapted workflow instance can be continued. Figure 8 shows the workflow monitoring tool after the dynamic adaptation. We remark that this example shows the usage of the system with respect to the graphical user interface and the implementation of the correctness check as described above.

5 Related Work

Flexibility is an important research topic in workflow management. This section reviews major approaches in this area, and it discusses how the work presented in this paper relates to these approaches. Finally, the workflow language of MQSeries Workflow, IBM's workflow management system is discussed.

The ADEPT project at the University of Ulm aims at providing flexible workflow support for advanced applications, mainly in the context of hospital environments [13]. Workflows are specified in the ADEPT workflow language, and dynamic modifications of workflows are modeled using ADEPT$_{flex}$, a formalism which consists of a set of rules, criteria, and methods which support the dynamic modification of workflow instances. Based on the ADEPT workflow language, a set of change operations are specified in the ADEPT$_{flex}$ framework [13]. In this framework, change operations to the structure of running workflows can be performed by users in a controlled manner. For instance, enhancing a workflow with a task involves the embedding of the added task into the workflow. The embedding of tasks in workflows is done by defining a set of tasks which have to be completed before the added one can start, and a set of tasks which can only be started after the added one has terminated. The conceptual work is validated by a prototypical implementation. The approach presented in this paper is more general than ADEPT, since no symmetric structure is imposed on workflow schemas. In addition, ADEPT does not provide workflow schema modification – the scope of each modification is a single workflow instance.

Flexibility issues in workflow management in general and workflow evolution in particular are investigated in [2] from a conceptual point of view, i.e., without a validation by an operational prototype. In that approach, workflow schemas consist of a set of tasks, which represent atomic workflow activities, i.e., workflow schemas are limited to two levels. There are control flow constraints, represented by directed

edges between task nodes as well as specific fork and join nodes, representing branching and joining of execution control, respectively. Data flow is not modeled explicitly. In contrast, each workflow has a set of workflow variables, which are global to that workflow. Flexibility is provided by workflow evolution. The rational behind workflow evolution is similar to dynamic adaptations: Given a workflow instance with a workflow schema, adapt (or migrate) the workflow instance to a modified workflow schema. However, the approach is different to the one reported in this paper. While the approach presented in [2] uses specific modification operations (called workflow evolution primitives) which are applied to the original workflow schema in order to modify it, the approach presented in this paper is more general, since a workflow instance can be adapted to an arbitrary workflow schema, provided the consistency criteria are fulfilled. While we propose consistency criteria based on the workflow instance and the new workflow schema, the notion of compliance is introduced in [2], which defines which workflow instances can be migrated to the new workflow schema version, created by modification operations.

Another interesting approach towards dynamic modifications of workflows is reported in [11]. The basic idea of that approach is to provide a rule-based mechanism which allows for a given semantic exception to compute the set of workflow instances and the required modifications of these instances to cope with the exceptional situation. However, the exceptions and the respective rules to cope with the exceptions have to be known at workflow modeling time. Once a semantic exception occurs, a rule-based system decides whether certain workflow activities can be dropped or whether new workflow activities have to be started. In a second step, the set of workflow instances which are affected by that exception are determined. In a third step, for each affected workflow instance the respective region where changes have to be performed is determined. Finally, workflow modifications are processed to adjust the workflow instance to the exceptional situation, and the modified workflow instance is continued.

The workflow language used in this paper is similar to the one in MQSeries Workflow [5], IBM's workflow management system. Workflow schemas are represented by directed graphs, consisting of workflow schema nodes, which can be related to each other by control flow and data flow. Instead of transition conditions used in MQSeries Workflow, our approach uses start conditions. Start conditions are more adequate when it comes to workflow schema re-use, since different occurrences of workflow schemas can have different start conditions. Moreover, a given start condition can be used in different workflow schemas. The formalization presented in Section 2 considers both workflow schemas and workflow instances in a single formalism; since MQSeries Workflow relies on a build-time versus run-time approach, workflow schemas and workflow instances are treated separately, and they are not represented in a single formalism. As indicated above, the integrated formal specification of workflow schemas and workflow instances is well suited to design and implement controlled dynamic adaptations of running workflow instances based on formal correctness criteria, a functionality not aimed at by MQSeries Workflow.

6 Conclusions

This paper introduces an approach to enhance the flexibility of workflow management systems by providing the ability to change the structure of workflow instances dynamically. By allowing to change the assignment of workflow instances and workflow schemas, different workflow schemas can be used to control a given workflow instance at different point in time. At each point in time, however, a workflow instance is attached to a single workflow schema. This elegant yet simple characterization allows to maintain the correctness property of workflow instances with respect to workflow schemas, while allowing dynamic adaptations.

Future work includes the use of compensating workflows to increase the number of workflow instances which can be adapted. Assume a workflow instance has proceeded to far to be adapted, by compensating, i.e., semantically undoing the workflows which do not have a corresponding sub-workflow schema in the new workflow schema, more workflow instances can be adapted dynamically. In addition, organizational issues like a concept for access control of dynamic adaptations, and its scope are in the focus of current and future work.

Acknowledgment: The author is grateful to the workflow group at the University of Münster for their involvement in designing and implementing the system.

References

[1] Bauzer Medeiros, C., Vossen, G., Weske, M.: *WASA: A Workflow-Based Architecture to Support Scientific Database Applications*. Revell, Tjoa (Editors): Proc. 6th DEXA Conference, Springer LNCS 978, pp 574–583. Berlin: Springer 1995

[2] Casati, F., Ceri, S., Pernici, B., Pozzi, G.: *Workflow Evolution*. Data and Knowledge Engineering Vol 24 No 3 1998. pp 211–238. Elsevier Science 1998

[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] Hündling, J.: *Development of a Graphical User Interface for a CORBA-Based, Flexible Workflow Management System*. (in German) Diplomhausarbeit. Institute for Information Systems, Universität Münster 1998

[5] IBM. *IBM MQSeries Workflow: Concepts and Architecture, Version 3.2*. Publ. No GH12-6285-01, 1999

[6] Iona Technologies: *OrbixWeb Programming Guide.* Dublin: Iona Technologies 1996

[7] Jablonski, S., Bußler, C.: *Workflow-Management: Modeling Concepts, Architecture and Implementation* International Thomson Computer Press, 1996

[8] 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)

[9] Kuropka, D.: *Specification and Implementation of a CORBA-Based Persistent Workflow Engine.* (in German) Diplomhausarbeit. Institute for Information Systems, Universität Münster 1998

[10] Leymann, F., Roller, D.: *Production Workflow: Concepts and Techniques*. Upper Saddle River: Prentice Hall 2000

[11] Müller, R., Rahm. E.: *Rule-Based Dynamic Modification of Workflows in a Medical Domain.* Buchmann (Editor): Proc. of BTW'99. Informatik Aktuell, pp 429–448. Berlin: Springer 1999

[12] Rational Software: *Unified Modeling Language – UML Notation Guide. Version 1.1*, September 1997 (available from www.rational.com/uml)

[13] 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

[14] Schuschel, H.: *Flexibility Issues in a Distributed, Flexible Workflow Management System.* (in German) Diplomhausarbeit. Institute for Information Systems, Universität Münster 1998

[15] Serries, T.: *Distribution and Scalability in CORBA-Based Workflow Management Systems.* (in German) Diplomhausarbeit. Institute for Information Systems, Universität Münster 1998

[16] 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

[17] 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

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

[19] Weske, M.: *Object-Oriented Design of a Flexible Workflow Management System*. Second East-European Symposium on Advances in Databases and Information Systems (ADBIS'98), Poznan, Poland. Springer Lecture Notes in Computer Science 1475, pp 119–130. Berlin: Springer 1998

[20] Weske, M.: *Foundation, Design, and Implementation of Dynamic Adaptations in a Workflow Management System.* Technical Report Angewandte Mathematik und Informatik 6/2000-I, University of Muenster, Germany 2000

[21] Weske, M., Kuropka, D.: *A Flexible Persistent Storage Service for Object-Oriented Middleware.* Technical Report Angewandte Mathematik und Informatik 5/2000-I, University of Muenster, Germany 2000