The WASA2 Object-Oriented Workflow Management System

Gottfried Vossen, Mathias Weske
University of Münster, Germany
http://wwwmath.uni-muenster.de/informatik/u/dbis/index.html

1 Introduction

Workflow management has gained increasing attention recently as an important technology to improve information system development in dynamic and distributed organizations. To develop a workflow application, selected business processes of an organization are modelled, optimized and specified as workflow schemas, using workflow languages [2]. Workflow schemas are used by workflow management systems to control the execution of workflow instances, i.e., representations of real-world business processes [3]. The first generation of workflow management systems (WFMS) were developed mainly to model and control the execution of business processes with fairly static structures, to be executed in homogeneous environments. Recently, the need for enhanced flexibility of workflow modeling and execution and the integration of applications in heterogeneous environments emerged in the workflow context [1]. The WASA project aims at supporting flexible and distributed workflows in heterogeneous environments [4]. This paper briefly overviews the conceptual design and implementation of the object-oriented workflow management system $WASA_2$, and sketches the proposed demo. References to work that relates to ours or that we started from are given in the cited WASA papers.

2 Design of the $WASA_2$ Workflow Management System

This section sketches the design goals of $WASA_2$, and it discusses its conceptual design by briefly presenting the $WASA_2$ workflow meta schema. We primarily focus the use of CORBA Common Object Services to implement the system.

In order to be suitable for a broad spectrum of workflow applications, there are different requirements that have to be met by a workflow management system. While there are numerous requirements of this kind, like the need for functional decomposition and modeling the technical and organization environment of workflow executions, we put our emphasis on the following:

- *Reuse of workflow schemas:* In order to allow efficient workflow modeling and to minimize redundancy in workflow modeling, it is important that workflow schemas are defined once to be used multiple times.
- *Integration:* An important functionality of a workflow management system is the integration of existing software systems in a single workflow application. A recent trend towards developing information systems is using business objects. A workflow management system should support the integration of business objects in workflow applications.
- *Flexibility:* Support for partially specified or dynamically changing application processes is not available in first generation workflow management systems. This limitation creates one of the main obstacles for the deployment of workflow applications, because workflow users and workflow administrators often encounter situations which require a dynamic change, i.e., a change of the workflow schema while the respective workflow runs. Dynamic changes allow rapid reactions to changes in the market environment, thus increasing the competitiveness of the organization.
- *Distribution and Scalability:* Traditionally, workflow management systems are client/server-based such that the server corresponds to a centralized workflow engine and the clients refer to the workflow users. Since workflow management is inherently performed in distributed settings, a centralized workflow engine can become a performance bottleneck of the system,

SIGMOD '99 Philadelphia PA

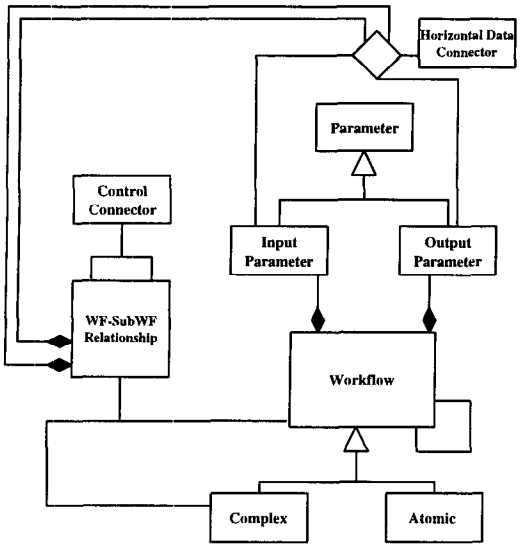

Figure 1: $WASA_2$ Workflow Meta Schema (simplified version).

and the availability of the workflow system then depends on the functionality of a single site.

- *Persistency:* The success of organizations nowadays relies to a considerable extent on the availability and fault tolerance of its information infrastructure. In particular, a system failure should not leave running workflow instances and data manipulated by them in an undefined state. In contrast, up-to-date information on the state of running workflow instances has to be stored in stable storage in order to be able to continue these workflows after a system restart. A key prerequisite to developing a functional workflow restart procedure is to maintain explicit state information of workflow instances and accompanying data in stable storage.

The $WASA_2$ *workflow meta schema* is shown in Figure 1 using the Uniform Modeling Language (UML). UML is an object-oriented modeling and design language, which we use for modeling workflow schemas and workflow instances; in UML, classes are represented by marked rectangles. In terms of terminology, that figure shows a class diagram of workflows and workflow-related entities and their relationships. From a workflow modeling point of view it shows a workflow meta schema, since it describes how workflow schemas (and workflow instances) are modelled.

The *Workflow* class is the central class of the $WASA_2$ workflow meta model. It contains workflow objects which are either workflow schema objects or workflow instance objects; we use the term *workflow* to indicate both workflow schemas and workflow instances. As described above, workflows can be either atomic or complex, and atomic workflows can be executed either automatically or manually. This structural property of workflows is reflected in the workflow meta schema by defining *Complex* workflow and *Atomic* workflow as sub-classes of class *Workflow* (and *Automatic* and *Manual* as sub-classes of the *Atomic* workflow class). The workflow hierarchy, i.e., the relationship between a complex workflow and its sub-workflows, is modeled by the *WF-SubWF Relationship* between a complex workflow and a (complex or atomic) workflow.

The software architecture is directly based on the $WASA_2$ workflow meta schema presented above. However, there are additional, implementation-specific properties, which are briefly discussed next. Since our system is based on a CORBA infrastructure, we make extensive use of Corba Common Object Services to implement the system; in particular, we have used OrbixWeb by Iona and Java Development Kit 1.1.6.

The overall design of the $WASA_2$ prototype is shown in Figure 2. The application level is the top level of the architecture. Users access workflow applications via the graphical user interface (GUI). To perform workflow activities, the GUI can start various applications, as defined in the workflow schema, e.g., office applications or existing domain-specific applications. We remark that the graphical user interface is configurable to support the needs of different user groups or tasks. In addition, there are integrated tools for the specification of workflow schemas and the monitoring and dynamic modification of workflow instances. The middle level comprises the facilities used to provide support for workflow applications. Due to the object-oriented approach we have taken, at this level there are workflow objects and related objects, as described in the $WASA_2$ workflow meta schema. In addition, business objects reside at the Facilities level. The data and functionality provided by business objects can be used by $WASA_2$ workflow object in an integrated fashion. As the system is based on CORBA, workflow objects, business objects, and the graphical user interface can communicate by the use of a CORBA Object Request Broker. To implement workflow objects and related objects, a set of CORBA Common Object Services can be used.

The *Persistency Service* is used to store arbitrarily structured objects, mainly workflow objects. This service provides transactional capabilities such as the well-known ACID properties and nested transactions. Workflow objects are generally persistent to provide a high degree of fault tolerance. In particular, when a workflow object which is currently down is sent a message, the ORB at the respective site consults the persistency service, which in turn reloads the requested workflow object into main memory and sends the request to that object.

The *Life Cycle Service* is used to create, copy, move,

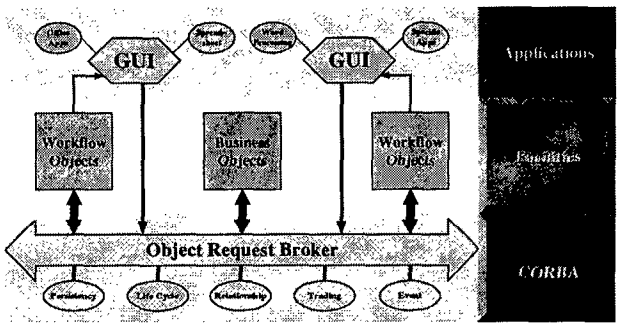

Figure 2: Architecture of $WASA_2$ system.

and delete CORBA objects. In $WASA_2$, this service is used to create workflow objects. In particular, a workflow schema object is created when a new workflow schema is defined during the workflow modeling phase. A workflow instance object is created whenever a workflow is instantiated, typically followed by an external event, for instance the receipt of an order in an order-processing workflow. Moving workflow objects between different sites of a CORBA environment is another function of the Life Cycle Service; this functionality is required whenever a workflow object has to be transferred to the location of a workflow execution.

There are complex relationships between different workflow objects and between workflow objects and related objects, for instance role objects or agent objects. Whenever a workflow instance is created, for instance, there is an instance-of relationship between the workflow instance object and the respective workflow schema object; this relationship is maintained by functionalities provided by the *Relationship Service*. This service allows to define role objects, i.e., objects which explicitly represent different roles a workflow objects can be in. Roles are a powerful mechanism to manage complex relationships, and roles are extensively used in the Corba Relationship Service.

In complex CORBA-based applications, objects with different properties reside in different locations of the CORBA environment. To support flexible queries in locating objects relative to specific properties, a trader is used. This component is implemented using the *Trading Object Service*. In $WASA_2$, this service is used to locate specific agents in distributed and complex organizations. This functionality can be utilized for flexible role resolution, which is based on the properties and availability of agents. The trader functionality is important since our approach is fully distributed, which implies the lack of a centralized agent that keeps track of the system state, or of the availability of agents capable to perform workflow activities.

The *Event Service* implements complex event notification mechanisms between CORBA objects. $WASA_2$ uses push communication, for instance, in workflow monitoring. Each workflow instance sends relevant events to an event channel, for instance the start of a sub-workflow instance. The workflow monitoring tool is a consumer of these events. When the start event of a sub-workflow instance occurs, the event is pushed to the workflow monitoring tool using the event channel and the functionality provided by the Event Service. On receiving the event, the workflow monitoring tool learns from the state change of the workflow instance it is monitoring.

3 About the Demo

The demo is based on a sample workflow schema, which can be instantiated multiple times to present different workflow instances. The graphical user interface of the $WASA_2$ system is able to display and to allow editing of workflow schemas. In addition, the progress of running workflow instances can be monitored using a coloring scheme which is based on states of workflow instances. To demonstrate the flexibility of the system, we present dynamic modifications of workflow instances, i.e., we show how the structure of a running workflow instance can be changed to allow to adapt workflow instances to changes in the environment of the workflow. The application we present is taken from the emerging domain of electronic commerce, where workflow management fits as an embedded technology.

References

[1] G. Vossen, M. Weske. *The WASA Approach to Workflow Management for Scientific Applications.* In: A. Dogac et al. (eds.): *Workflow Management Systems and Interoperability.* ASI NATO Series F, Vol. 164, pp. 145-164. Berlin: Springer 1998

[2] M. Weske, G. Vossen. *Workflow Languages.* In: P. Bernus et al. (eds.): *Handbook on Architectures of Information Systems*, pp. 359-379. Berlin: Springer 1998

[3] Weske, M., T. Goesmann, R. Holten, R. Striemer. *A Reference Model for Workflow Application Development Processes.* In Proc. Int. Joint Conf. on Work Activities Coordination and Collaboration (WACC), San Francisco, February 1999, pp. 1–10

[4] M. Weske. *Workflow Management Through Distributed and Persistent CORBA Workflow Objects.* To appear in Proc. 11th Int. Conf. on Advanced Information Systems Engineering (CAiSE), Heidelberg, Germany, June 1999