

Workflow Management Through Distributed and Persistent CORBA Workflow Objects

Mathias Weske

Lehrstuhl für Informatik, Universität Münster
Steinfurter Straße 107, D-48149 Münster, Germany
weske@helios.uni-muenster.de

1 Introduction

Workflow management has gained increasing attention recently as an important technology to improve information system development in flexible and distributed organizations [2, 3, 8, 5]. The WASA project aims at supporting flexible and distributed workflows, using an object-oriented design and the CORBA middleware standard. This paper describes major design goals of WASA₂, and it sketches the architecture and implementation of the system based on CORBA workflow objects and CORBA Common Object Services.

2 WASA₂ Design Goals

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 [6]; we put our emphasis on the topics distribution and scalability, persistency, and flexibility; additional topics are discussed in [9].

Distribution and Scalability Traditionally, workflow management systems are client/server systems such that the server corresponds to a workflow engine, and workflow users access the system using workflow clients as interfaces. Considering workflow executions, this approach can be conceived as distributed since the work performed during the workflow is done in the client side, and typically there are multiple clients involved in a workflow execution. However, as far as controlling the execution of workflows is concerned, the client/server-approach is in fact a centralized one: The workflow engine is a centralized server, which controls the execution of all workflows in a given organization. In large scale workflow applications, the centralized site is likely to receive a heavy load, which may result in performance problems and scalability issues. Besides the fact that the centralized workflow engine can become a performance bottleneck, it is a single point of failure, such that the availability of the workflow application depends on a single site. Regarding performance and fault tolerance issues, for these kind of application it would be much more adequate to use a distributed approach instead of a central one. Thereby we mean that workflow control is performed decentralized, not just workflow executions as is done in today's workflow management systems.

Persistency The success of organizations nowadays relies to a considerable extent on the availability of its information infrastructure. In the database context, this has led to the development of elaborate concurrency control and recovery techniques, aiming at providing execution guarantees in the presence of multiple concurrent users and system failures. Providing this kind of functionality is also important for workflow management systems. In particular, a system failure should not leave running workflow instances and manipulated data 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 interrupted workflows after the system is restarted. A key prerequisite to develop a fault-tolerant workflow management system which implements a functional workflow restart procedure is to maintain explicit state information of workflow instances and accompanying data on running workflow executions in stable, persistent storage.

Flexibility The first generation of workflow management systems are well suited for modeling and controlling the execution of application processes with a pre-defined and static structure [3, 2]. Adequate support for partially specified or dynamically changing application processes, however, is not provided. This limitation is seen today as one of the main obstacles for the deployment of workflow applications in real-world organization, because workflow users and workflow administrators often encounter situations which have not been foreseen at workflow modeling time and which thus require a dynamic change. As a result, a desirable feature of workflow management systems is to allow structural changes of workflows while they are executing [5, 6]; this functionality is known as dynamic change. Allowing rapid changes of workflows increases the competitiveness of an organization significantly.

3 Systems Design

The architecture of the WASA₂ prototype consists of three levels (Figure 1): The application level is the top level of the architecture. Users access workflow applications with a graphical user interface (GUI). To perform workflow activities, the GUI can start external applications, as defined in the workflow schema, for instance office applications, existing domain-specific applications or methods provided by business objects. We remark that the graphical user interface is configurable to support the needs of different user groups. In addition, there are integrated tools for the specification of workflow schemas and for monitoring and dynamic modification of workflow instances. In the intermediate level there are facilities used to provide support for workflow applications. Due to the object-oriented approach, on this level there are workflow objects and related objects, as described in the WASA₂ workflow meta schema [9]. Business objects also reside on the facility level. Workflow objects, business objects and the graphical user interface communicate through a CORBA Object Request Broker. To implement workflow objects and related objects, CORBA Common

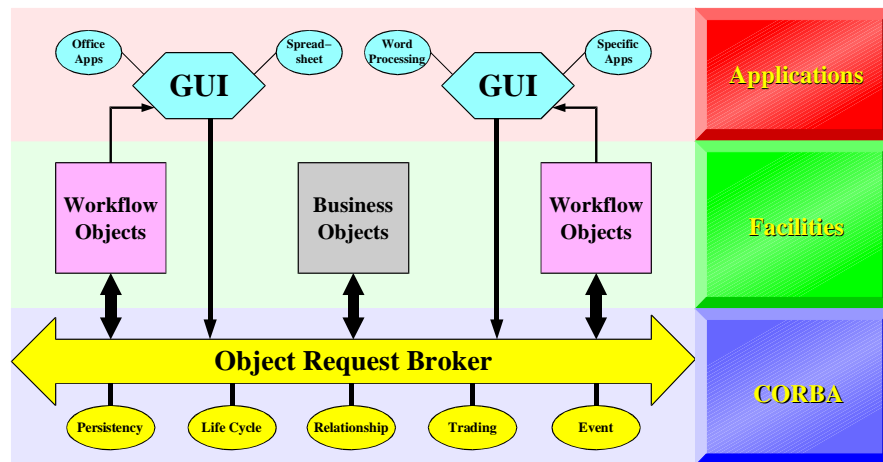


Fig. 1. Architecture of WASA₂ system.

Object Services (COS) are used extensively; in the WASA project, we have implemented a number of CORBA COS. Now we describe how the design goals are supported by the WASA₂ system.

Distribution and Scalability In WASA₂, workflows are executed and controlled fully distributed, using CORBA workflow objects: The execution of a complex workflow starts with the initiation of its sub-workflows. Without going into the technical details, sub-workflow instances are instantiated as CORBA objects, and the control flow and data flow connectors between them are also represented by CORBA objects. The execution is controlled by messages, sent between workflow objects. In particular, to start a sub-workflow, the super-workflow sends a message to that sub-workflow instance. On receipt of that message, the sub-workflow is started. When the sub-workflow instance terminates, it sends a message to each workflow instance, which follows the terminated workflow in control flow, as defined by the respective workflow schema. This procedure continues until the last sub-workflow instance of the complex workflow terminates; this workflow sends a message to the super-workflow. On receipt of that message, the complex workflow is terminated. We mention that workflow objects can reside in different sites of a distributed computing system. By placing workflow objects on different sites (and by adding machines if necessary), the system scales up nicely. The Internet Inter ORB protocol (IIOP) allows the transparent communication of multiple ORBs.

Persistency Persistent workflow objects is the prerequisite for fault tolerant workflow executions. In WASA₂, a Persistency Service is used to implement persistent workflow objects. Assume there is a complex workflow instance i with sub-workflow instances j, k, l , to be executed sequentially: i sends a message to j and waits for the message from l , indicating the successful completion of the complex workflow. Assume the site of i goes down after it sent a start-message to j , and the other workflow instances are in different sites. Then the sub-workflows j, k and l can execute, even though i is currently not available. When l terminates it tries to send a message to i , informing i of its completion. Assuming the site of i is up again at this point in time, the ORB daemon in that site accepts the request. Since the object i is not in main memory, the ORB daemon loads it dynamically from persistent storage using the Persistency Service. Notice that the state of i is valid since the transactional property of the Persistency Services guarantees that i is in a consistent state. After the complex workflow instance i is loaded, the message is sent to i , which resumes execution.

Flexibility WASA₂ supports dynamic changes, i.e., the ability to change workflow instances while they execute. As is elaborated in [9], the system determines the set of workflow instances which can be adapted to a particular dynamic change, and it allows to continue workflow instances with the changed workflow schema by changing the instance-of relationship between workflow instance objects and workflow schema objects. We remark that dynamic changes have to be embedded in an organizational framework, which defines who has the ability and competence to change workflow schemas and running workflow instances. The WASA₂ system, however, provides the mechanisms to support dynamic modifications on the technical level.

4 Conclusions

Our work is related to the quest of the OMG to define a Workflow Management Facility. However, the Workflow Management Facility proposal [1] stresses runtime interoperability of existing workflow management systems and the ability to use workflow monitoring and auditing tools in these settings [6]. Other approaches to object-oriented workflow management system design include the Meteor₂ project at the University of Georgia [7]. In Meteor₂, workflow specifications are translated into executable code, which is then executed in a distributed fashion. CORBA is used in the Meteor₂ system as a communication infrastructure for distributed workflow executions.

This paper overviews the conceptual foundations and the system design and implementation of WASA₂, an object-oriented workflow management system based on CORBA. An in-depth presentation of the conceptual design and implementation of WASA₂ can be found in [9]. Key features are reuse of workflow schemas, distributed workflow execution control, persistent workflow executions, and the support for dynamic modifications of running workflow instances with the ability to control the scope of these changes. It is interesting to notice that

the requirements as specified in [6] to a large extent are satisfied by the WASA₂ system, involving (i) changes of the underlying process model, (ii) composition of reusable business components, (iii) monitoring of process execution, (iv) distribution of a process across business domains and (v) assignment of process steps to workflow participants.

Acknowledgments: The author is thankful to the WASA group at the University of Münster.

References

1. CoCreate Software, Conventus, CSE Systems, Data Access Technologies, Digital Equipment Corp., DSTC, EDS, FileNet Corp., Fujitsu Ltd., Hitachi Ltd., Genesis Development Corp., IBM Corp., ICL Enterprises, NIIP Consortium, Oracle Corp., Plexus - Division of BankTec, Siemens Nixdorf Informationssysteme, SSA, Xerox: *BODTF-RFP 2 Submission Workflow Management Facility (jointFlow)*. OMG Document bom/98-06-07 (1998)
2. Georgakopoulos, D., Hornick, M., Sheth, A.: *An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure*. Distributed and Parallel Databases, 3:119–153, 1995
3. Leymann, F., Altenhuber, W.: *Managing Business Processes as an Information Resource*. IBM Systems Journal 33, 1994, 326–347
4. OMG: *CORBAServices: Common Object Services Specification*. (available from www.omg.org)
5. Reichert, M., Dadam, P.: *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
6. Schmidt, M.-T.: *Building Workflow Business Objects*. OOPSLA'98 Workshop 8: Business Object Design and Implementation IV: From Business Objects to Complex Adaptive Systems (download from www.jeffsutherland.org/oopsla98/mts.html on 11-11-98)
7. Sheth, A., Kochut, K.J.: *Workflow Applications to Research Agenda: Scalable and Dynamic Work Coordination and Collaboration Systems*. In: Dogac, Kalinichenko, zsu, Sheth (Eds.): *Workflow Management Systems and Interoperability*. NATO ASI Series, Series F: Computer and Systems Sciences, Vol. 164, 35–60. Berlin: Springer 1998
8. Vossen, G., Weske M.: *The WASA Approach to Workflow Management for Scientific Applications*. In: Dogac, Kalinichenko, zsu, Sheth (Eds.): *Workflow Management Systems and Interoperability*. NATO ASI Series, Series F: Computer and Systems Sciences, Vol. 164, 145–164. Berlin: Springer 1998
9. Weske, M.: *Design and Implementation of an Object-Oriented Workflow Management System based on CORBA*. Fachbericht Angewandte Informatik und Mathematik 33/98-I, University of Muenster 1998