ADEPT$_{workflow}$ - Advanced Workflow Technology for the Efficient Support of Adaptive, Enterprise-wide Processes

Clemens Hensinger, Manfred Reichert, Thomas Bauer, Thomas Strzeletz, Peter Dadam
Dept. Databases and Information Systems, University of Ulm, Germany
{hensinger, reichert, bauer, strzeletz, dadam}@informatik.uni-ulm.de

1 The ADEPT Project

The ADEPT[1] project started in 1994 aiming at the development of an adaptive, process-oriented WfMS for (potentially) enterprise-wide application scenarios. It concentrates on the realization of flexible, robust WF-based applications, which have to run stable and which are used by non-computer-experts. To achieve the desired robustness, an appropriate formal WF model has been developed which allows to validate the correctness of a WF template already at build-time (e.g., exclusion of deadlocks, proper invocation of activity components, etc.). For achieving flexibility, appropriate concepts have been developed to support ad-hoc changes of in-progress WF (e.g., to add new steps, to omit steps, to change the sequence of steps, or to jump forward/backward in the flow), while maintaining their correctness and consistency [1]. Such deviations from the pre-modeled WF template may become necessary to cope with exceptional situations or to model incompletely defined parts of a WF template at run-time [2]. To ensure that the offered change facilities are usable by "normal" WF participants, high-level, powerful modification operations have been developed, which hide the complexity of a dynamic change to a large degree from the user. That means, that they are not burdened with the re-mapping of activity parameters, the problem of missing input values due to the deletion of a step, or the avoidance of any obscure system behavior (e.g., deadlocks). Instead users may define a dynamic change at a high semantic level (e.g., to insert a new step between two node sets), without requiring that they are familiar with a WF editing tool or with a WF description formalism.

Another important issue in the ADEPT project is the handling of temporal aspects. In the context of flexibility, for example, a dynamic change within an iterative process ("loop") may be only valid for one iteration or also for subsequent iterations. With respect to temporal constraints we came to the conclusion, that supervision of deadlines solely is not sufficient in many cases. We, therefore, developed additional concepts to supervise minimal and maximal time distances between activities, if specified. The system monitors these time constraints and analyzes the consequences with respect to subsequent steps; e.g., time inconsistencies after inserting a new activity.

As indicated above, we are interested in the support of large-scale enterprise-wide WF applications. In such scenarios, performance is a critical issue. Due to the high amount of communication between WF server(s) and WF clients, the communication network may become a bottleneck, especially if a large amount of "long-distance" communication occurs. To avoid bottlenecks, in the ADEPT project we have developed appropriate concepts to reduce the network load by migrating the control of a WF instance from one WF server to another at run-time, if this is favorable [3, 4].

2 The ADEPT$_{workflow}$ Prototype

Since 1997, we have developed a WFMS based on the concepts described above. The current version does already support many of these advanced features; e.g., ad-hoc deviations, time management, and distributed control of WF instances. At the conference, we demonstrate, how run-time modifications (deletion and insertion of WF steps) are specified and executed, how process migrations between different *ADEPT$_{workflow}$ servers* are performed, and how time constraints (e.g., time distances) are supervised. For demonstration purposes, we use the *ADEPT$_{workflow}$ demo client*, which does not only show worklists, but also visualizes the WF instance graphs (e.g., after a dynamic modification or a process migration has been carried out). Besides these run-time aspects, we give insights into the build-time components of the ADEPT$_{workflow}$ prototype: Process templates can be defined with the syntax-driven *ADEPT$_{workflow}$ modeler*, which supports numerous correctness and consistency checks (e.g., with respect to data flow). Complex organizational entities and relationships are managed by the *ADEPT$_{workflow}$ organization tool*.

[1] ADEPT stands for Application Development based on Encapsulated Pre-modeled Process Templates

3 System Architecture

The *ADEPT*$_{workflow}$ prototype uses a multi-server-architecture (cf. Fig. 1). Different clients may be connected to each WF server; e.g., worklist handlers, monitoring components, or tools for the definition of WF templates and organization models. For the implementation of such clients, ADEPT offers a rich API with a functionality comparable to that of the WfMC standard interfaces and with important extensions for the provision of the advanced ADEPT features. We have also extended the one-directional client-server-communication in order to support more advanced concepts (e.g., to get approvals from WF participants for performing a run-time deviation or to immediately notify them afterwards). A WF server may play an active role by initiating requests to clients. This has required extensions of the ADEPT API and therefore influences the way client programs have to be implemented.

The WF server itself is implemented on top of a RDBMS. This enables the transactional execution of requests and, therefore, guarantees the persistency and consistency of the WF data. The kernel of the server is realized as a multi-layer architecture in order to increase its adaptability and expandability. The top level, the **Execution Layer**, processes client API calls. Each call is decomposed into a set of service requests from the underlying **Service Layer**. This layer comprises services, which have been designed along the different WF aspects, like e.g., the management of process templates, process instances, worklists, organizational entities, or temporal constraints. As an example, consider the completion of a WF step by a client. This leads to an update of the time schedule and state of the WF instance, a role resolution of subsequent steps, and an update of worklists. The Service Layer may be easily extended by adding new components (e.g., for handling inter-workflow-dependencies). Each component of this layer decomposes a call into several basic operations for the **Data Access Layer** (e.g., to read, to create, or to modify WF objects). If a migration of the WF control or a synchronization of the WF data becomes necessary, in addition, the **Distribution Layer** provides the required data and performs the migration. At the receiving site, the necessary data access operations are performed. Finally, the Data Access Layer translates these basic operations into database calls for the underlying DBMS. – All components of the ADEPT$_{workflow}$ prototype are implemented in Java; for communication Java-RMI is used.

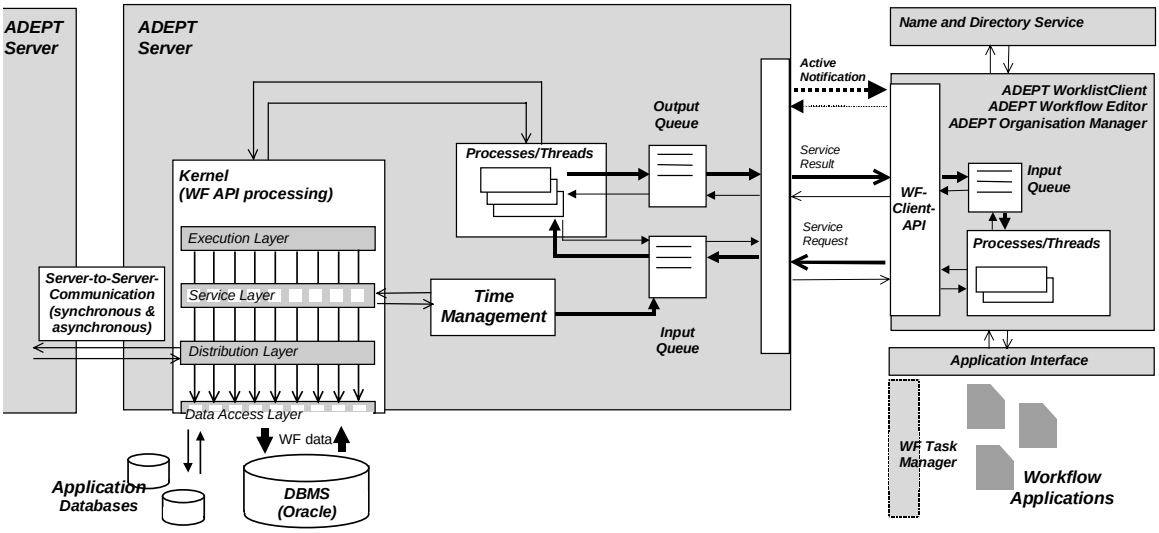

Figure 1: *ADEPT*$_{workflow}$ *System Architecture*

References

[1] M. Reichert and P. Dadam. ADEPT$_{flex}$ – Supporting Dynamic Changes of Workflows Without Losing Control. *Journal of Intelligent Information Systems*, Vol. 10, No. 2, March 1998, pp. 93-129.

[2] M. Reichert, C. Hensinger, and P. Dadam. Supporting Adaptive Workflows in Advanced Application Environments. *Proc. of the EDBT Workshop on Workflow Management Systems*, Valencia, Spain, March 1998, pp. 100-109.

[3] T. Bauer and P. Dadam. A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration. *Proc. 2nd IFCIS Conf. on Coop. Inf. Sys.*, Kiawah Island, SC, June 1997, pp. 99-108.

[4] T. Bauer and P. Dadam. Efficient Distributed Control of Enterprise-Wide and Cross-Enterprise Workflows. *Proc. Workshop on Enterprise-wide and Cross-enterprise Workflow Management* (Informatik '99), Paderborn, Ulmer Informatik-Bericht 99-07, University of Ulm, October 1999, pp. 25-32.