Mentor-lite Customizability: Tailoring a Light-Weight Workflow Management System to Workflow Application and Organizational Needs

Michael Gillmann, Jeanine Weissenfels, German Shegalov, Wolfgang Wonner, Gerhard Weikum
Department of Computer Science
University of the Saarland, Germany
WWW: http://www-dbs.cs.uni-sb.de
E-mail: {gillmann,weissenfels,shegalov,wonner,weikum}@cs.uni-sb.de

Abstract

The Mentor-lite prototype has been developed within the research project "Architecture, Configuration, and Administration of Large Workflow Management Systems" funded by the German Science Foundation (DFG). A salient feature of Mentor-lite is its ability to customize its workflow administration capabilities like worklist management and history management to the specific needs of an application and the organization of the underlying enterprise(s). The demo will show the feasibility of the presented approach by demonstrating the tailoring of worklist management policies.

1 System Overview

The Mentor-lite prototype has evolved from the Mentor workflow management system [MWW+98a, MWW+98b], but aims at a simpler architecture. The main goal of Mentor-lite has been to build a light-weight, extensible, and tailorable system with small footprint and easy-to-use administration capabilities. Our approach is to provide only kernel functionality inside the workflow engine, and consider system components like history management and worklist management as extensions on top of the kernel. The key point to retain the light-weight nature is that these extensions are implemented as workflows themselves. An invocation interface for application programs is provided by a generic IDL interface on the engine side and specific *wrappers* on the application side [MWG+99].

As shown in Figure1, the basic building block of Mentor-lite is an *interpreter* for workflow specifications. In Mentor-lite, workflows are specified in terms of state and activity charts, the specification formalism that has been adopted for the behavioral dimension of the UML industry standard and was already used in Mentor. Two additional components, the *communication manager* (*ComMgr*) and the *log manager* (*LogMgr*), are closely integrated with the workflow interpreter. All three components together form the *workflow engine*. The execution of a workflow instance can be distributed over several workflow engines at different sites. A separate *workflow log* is used at each site where a Mentor-lite workflow engine is running. Databases like the *workflow repository* (i.e., a repository of workflow specifications) or the *worklist database* can be shared by Mentor-lite workflow engines at different sites.

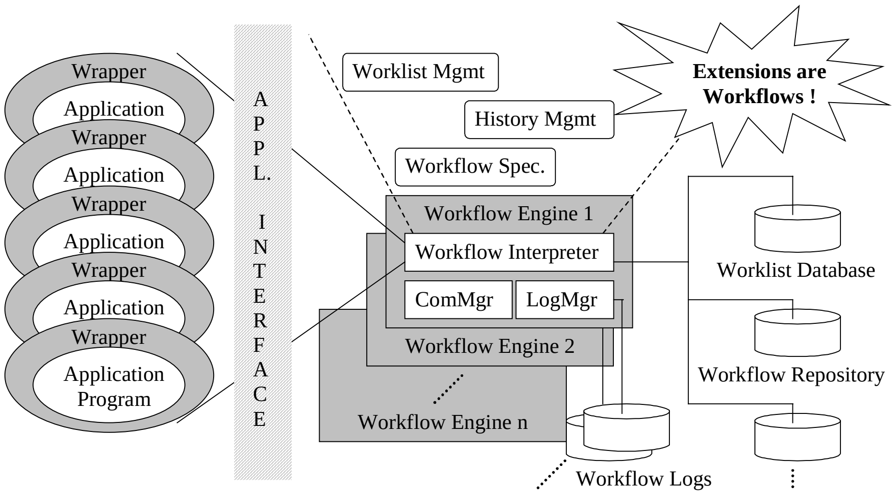

Figure 1 : The Mentor-lite architecture

2 Implementation Details

The workflow specifications are interpreted at runtime, which is a crucial prerequisite for flexible exception handling and dynamic modifications during runtime. The interpreter performs a stepwise execution of the workflow specification according to its formal semantics [WW97]. For each step, the activities to be performed by the step are determined and started.

Application dependent facilities like *worklist management* are implemented on top of the engine as state and activity charts. Hence, they are interpreted by the workflow interpreter just like any other workflow specification. This allows us to use the functionality of the Mentor-lite workflow engine through a single interface, namely the state chart and activity chart interpreter. The activities used in these subworkflows store worklist data in an Oracle database. The user interfaces, i.e. the worklists are implemented as Java applets. The applets use the JDBC interface to access the databases.

3 About the Demo

For the demo, we use the specification of a simple e-commerce workflow that we developed for benchmarking workflow management systems [GMW+99]. The workflow builds on the TPC-C benchmark for transaction systems, but enhances it by control and data flow between the activities "NewOrder", "Shipment", and "Payment" and includes additional activities. We will show several strategies for managing the users' worklists, specified in the form of state and activity charts. An example is given in Figure 2, which shows the specification of a load balancing workitem assignement for the "NewOrder" activity.

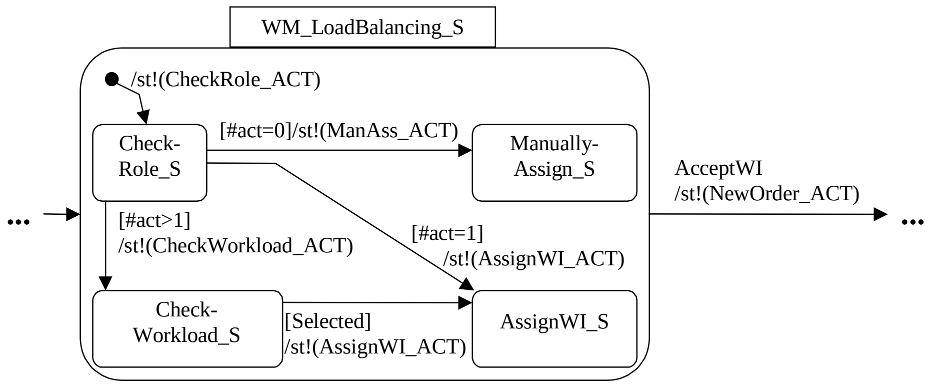

Figure 2 : Example of specifying a worklist management policy

References

[DKO+98] A. Dogac, L. Kalinichenko, M. Tamer Ozsu, A. Sheth (Eds.), Workflow Management Systems and Interoperability, NATO Advanced Study Institute, Springer, 1998

[GMW+99] M. Gillmann, P. Muth, G. Weikum, J. Weissenfels, Benchmarking of Workflow Management Systems (in German), German Conf. on Databases in Office, Engineering, and Scientific Applications (BTW), Freiburg, Germany, 1999

[MWG+99] P. Muth, J. Weissenfels, M. Gillmann, G. Weikum, Integrating Light-Weight Workflow Management Systems within Existing Business Environments, Int'l Conf. on Data Engineering (ICDE), Sydney, Australia, 1999

[MWW+98a] P. Muth, D. Wodtke, J. Weissenfels, G. Weikum, A. Kotz Dittrich, Enterprise-wide Workflow Management based on State and Activity Charts, in [DKO+98]

[MWW+98b] P. Muth, D. Wodtke, J. Weissenfels, A. Kotz Dittrich, G. Weikum, From Centralized Workflow Specification to Distributed Workflow Execution, Journal of Intelligent Information Systems, Special Issue on Workflow Management, Vol. 10, No. 2, 1998

[WW97] D. Wodtke, G.Weikum, A Formal Foundation for Distributed Workflow Execution Based on State Charts, Int'l Conf. on Database Theory (ICDT), Delphi, Greece, 1997