A Goal-driven Auto-Configuration Tool for the Distributed Workflow Management System Mentor-lite

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

Paper ID: D24

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). In this paper, we outline the distributed architecture of Mentor-lite and elaborate on a goal-driven auto-configuration tool for Mentor-lite and similar workflow management systems (WFMS). This tool aims to recommend an appropriate system configuration in terms of replicated workflow, application, and communication servers, so as to meet given goals for performance, availability, and performability at low system costs. The demo will show the monitoring capabilities of Mentor-lite and the various components of the auto-configuration tool.

1 System Overview

The Mentor-lite prototype system for distributed workflow management has evolved from its predecessor Mentor [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 workflow management system (WFMS) 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.

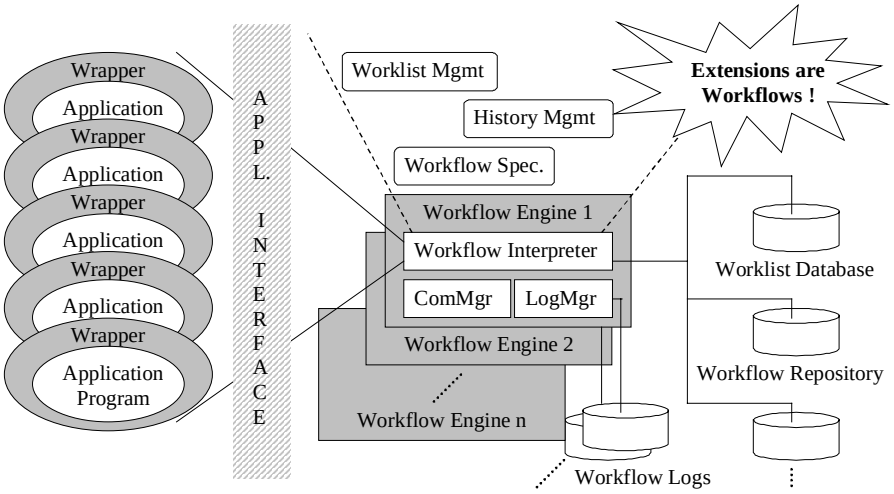

Figure 1 : The Mentor-lite architecture

2 Distributed Workflow Execution

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.

Mentor-lite supports a protocol for distributed execution of workflows spread across multiple workflow engines. This support is crucial for workflows that span large, decentralized enterprises with largely autonomous organizational units or even cross multiple enterprises to form so-called “virtual enterprises”. The communication manager is responsible for sending and receiving synchronization messages between the engines. These messages contain information about locally raised events, updates of state chart variables and state information of the local engine [MWW+98b]. When a synchronization message is received, the corresponding updates at the receiving site are performed. In order to guarantee a consistent global state even in the presence of site or network failures, we have built reliable message queues using the CORBA Object Transaction Services. As the basic communication infrastructure for the distributed execution we use the CORBA implementation Orbix.

The use of an object request broker (ORB) for the communication layer allows us not only to distribute the workflow execution, but also to seperate the execution of the applications from the workflow engines. Consequently, we are able to run the applications on dedicated application servers that can be replicated for availability and performance reasons. Finally, CORBA provides the replication of the communication servers, i.e., the ORBs, itself.

3 Workflow Monitoring

For administration, Mentor-lite provides a Java-based workbench for workflow design, workflow partitioning across multiple workflow servers, and a Java-based runtime monitoring tool illustrated in Figure 2. The monitoring tool is able to display the current execution state of a running workflow, highlighting the control flow path traversed so far (see Figure 2: the current state is colored red and the traversed control flow is blue). Further runtime data such as the current values of control flow variables are also available. In addition to tracking and visualizing the execution of individual workflow instances, the monitoring tool can also provide aggregate figures about workflow types, e.g., the invocation rate of workflows, the frequency of specific activities within workflows of a given type, etc.

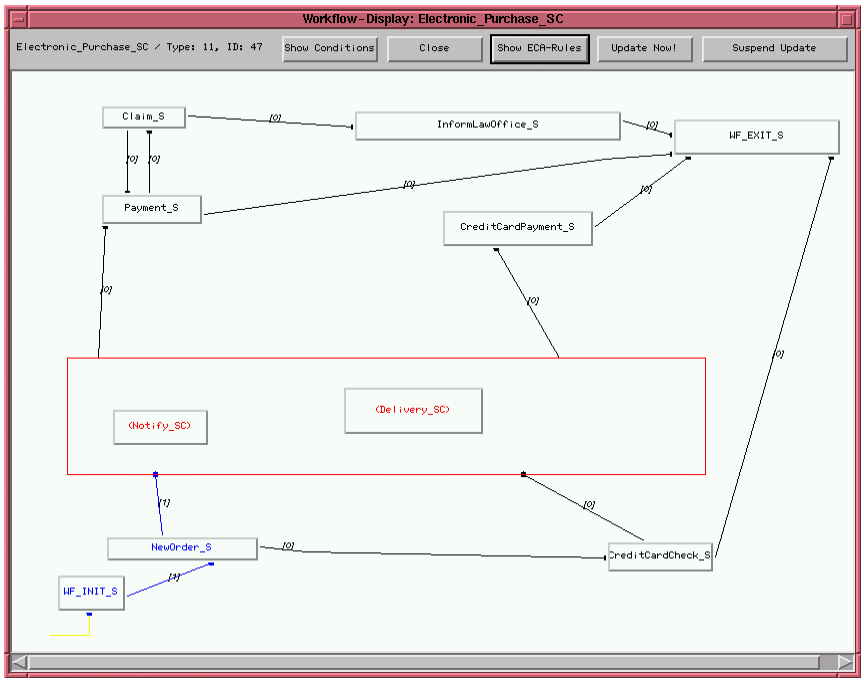

Figure 2 : Monitoring of a workflow instance

4 The Auto-Configuration Tool

A distributed configuration of Mentor-lite consists of different workflow servers (i.e., instances of the workflow engine), application servers, and one communication server (i.e., ORB). Each server of the first two categories can be dedicated to a specified set of workflow activities and invoked applications, resp., on a per type basis. Each of these dedicated servers and also the communication server can be replicated across multiple computers for enhanced performance and availability. Given this flexibility (which is provided in similar ways also by some commercial WFMSs), it is a difficult problem to choose an appropriate configuration for the entire WFMS that meets all requirements with regard to throughput, interaction response time, and availability. Moreover, it may be necessary to adapt an initial configuration over time due to changes of the workflow load, e.g., upon adding new workflow types.

To this end, we have developed a suite of analytic models, using stochastic methods like continuous-time Markov chains and Markov reward models, to predict the performance, availability, and performability under a given load. The performance model estimates the maximum sustainable throughput in terms of workflow instances per time unit and the mean waiting time for service requests such as interactions upon starting an activity on the basis of a Markov chain model for the statistical behavior of the various workflow types. The availability model estimates the mean downtime of the entire system for given failure and restart rates for the various components. Finally, the performability model takes into account the performance degradation during transient failures and estimates the effective mean waiting time for service requests with explicit consideration of periods during which only a subset of a server type's replicas are running. These models, which are described in detail in [GWW+00], form the underpinnings of an auto-configuration tool for distributed WFMSs.

The auto-configuration tool is primarily driven by statistics on the workload from the monitoring tool of Mentor-lite. It can feed this information into its analytic models to a hypothetical configuration in a what-if analysis. By systematic variation of the parameters for such hypothetical configurations the tool is also able to derive the (analytically) best configuration, i.e., the minimum degree of replication of each of the involved server types to meet given availability and performance or performability goals, and recommend appropriate reconfigurations. The tool is largely independent of a specific WFMS, using product-specific stubs for its various components that need to interact with the WFMS.

The components of the configuration tool and its embedding into the overall system environment are illustrated in Figure 3. The tool consists of four main components:

- the *mapping* of workflow specifications onto the tool's internal models,
- the *calibration* of the internal models by means of statistics from monitoring the system,
- the *evaluation* of the models for given input parameters, and
- the computation of *recommendations* to system administrators and architects, with regard to specified goals.

For the mapping the tool interacts with a workflow repository where the specifications of the various workflow types are stored. In addition, statistics from online monitoring are used as a second source (e.g., to estimate typical control flow behavior etc.). The configuration tool translates the workflow specifications into corresponding continuous-time Markov chain models. For the evaluation of the models, additional parameters may have to be calibrated; for example, the first two moments of server-type-specific service times for various elementary service requests (e.g., starting an activity) have to be fed into the models. This calibration is again based on appropriate online monitoring. So both the mapping and calibration components exploit online statistics about the running system. Consequently, when the tool is to be used for configuring a completely new workflow environment, many parameters have to be intellectually estimated by a human expert. Later, after the system has been operational for a while, these parameters can be automatically adjusted, and the tool can then make appropriate recommendations for reconfiguring the system.

The evaluation of the tool's internal models is primarily driven by specified performance goals with consideration of transient component downtimes, so-called *performability goals*. System administrators or architects can specify goals of the following kinds:

- the minimum throughput in terms of completed workflow instances per time unit that the entire WFMS must be able to sustain,
- a tolerance threshold for the mean waiting time of service requests that would still be acceptable to the end-users,

- a tolerance threshold for the unavailability of the entire WFMS, or in other words, a minimum availability level.

The first two goals requires evaluating the performance or the more comprehensive performability model, whereas the third one merely needs the availability model. The tool can invoke these evaluations either for a given system, or it can search for the minimum-cost configuration that satisfies all goals. The cost of a configuration is assumed to be proportional to the total number of servers that constitute the entire WFMS, but this could be further refined with respect to different server types. Also, the goals can be refined into workflow-type-specific goals, by requiring, for example, different maximum waiting times or availability levels for specific types.

The tool uses the results of the model evaluations to generate recommendations to the system administrators or architects. Such recommendations may be asked for regarding specific aspects only (e.g., focusing on performance and disregarding availability), and they can take into account specific constraints such as limiting or fixing the degree of replication of particular server types (e.g., for cost reasons).

The most far-reaching use of the configuration tool is to ask it for the minimum-cost configuration that meets specified performability and availability goals. Computing this configuration requires searching the space of possible configurations, and evaluating the tool's internal models for each candidate configuration. While this may eventually entail full-fledged algorithms for mathematical optimization such as branch-and-bound or simulated annealing, our first version of the tool uses a simple greedy heuristics. The algorithm iterates over candidate configurations by increasing the number of replicas of the most critical server type until both the performability and the availability goals are satisfied. Since either of the two criteria may be the critical one and because an additional server replica improves both metrics at the same time, the two criteria are considered in an interleaved manner. Thus, each iteration of the loop over candidate configurations evaluates the performability and the availability, but adds servers to two different server types only after re-evaluating whether the goals are still not met. This way the algorithm avoids "oversizing" the system configuration.

To summarize, the functionality of the configuration tool comprises an entire spectrum ranging from the mere analysis and assessment of an operational system all the way to providing assistance in designing a reasonable initial system configuration, and, as the ultimate step, automatically recommending a reconfiguration of a running WFMS.

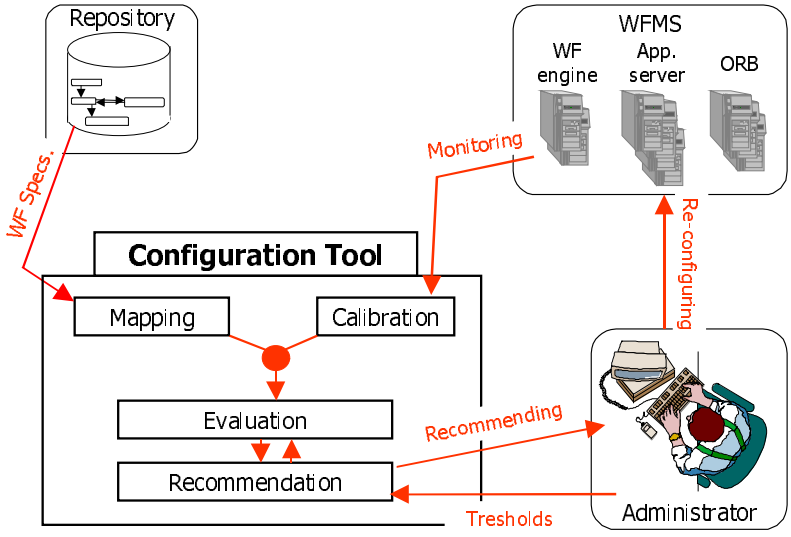

Figure 3 : Integration of the auto-configuration tool

5 About the Demo

The demo consists of two parts. The first part shows the distributed architecture of the Mentor-lite prototype. The second part shows the various components of the auto-configuration tool and the tool's interaction with Mentor-lite.

For the demo of the Mentor-lite prototype we use the specification of a simple e-commerce workflow that we developed for the benchmarking of WFMSs [GMW+99]. To illustrate the complete functionality of control flow handling, the workflow includes the full spectrum of control flow structures, i.e, splits, parallelism, joins, and loops. 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. For illustration, Figure 4 shows two kinds of user interfaces, the worklist of user John Doe and the activity "NewOrder" which corresponds to the work item "insert new order" in the worklist. The worklist presents all pending work items assigned to the user John Doe.

The demo of the auto-configuration tool will show what-if analyses for assessing given hypothetical configurations, and it will demonstrate the steps of the iterated model evaluations that are needed for deriving the analytically best configuration for given performability and availability goals. Both of these different uses of the tool will highlight the interactions with the entire WFMS, especially the monitoring component.

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

[GWW+00] M. Gillmann, J. Weissenfels, G. Weikum, A. Kraiss, Performance and Availability Assessment for the Configuration of Distributed Workflow Management Systems, Conf. on Extending Database Technology (EDBT), Konstanz, Germany, 2000

[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

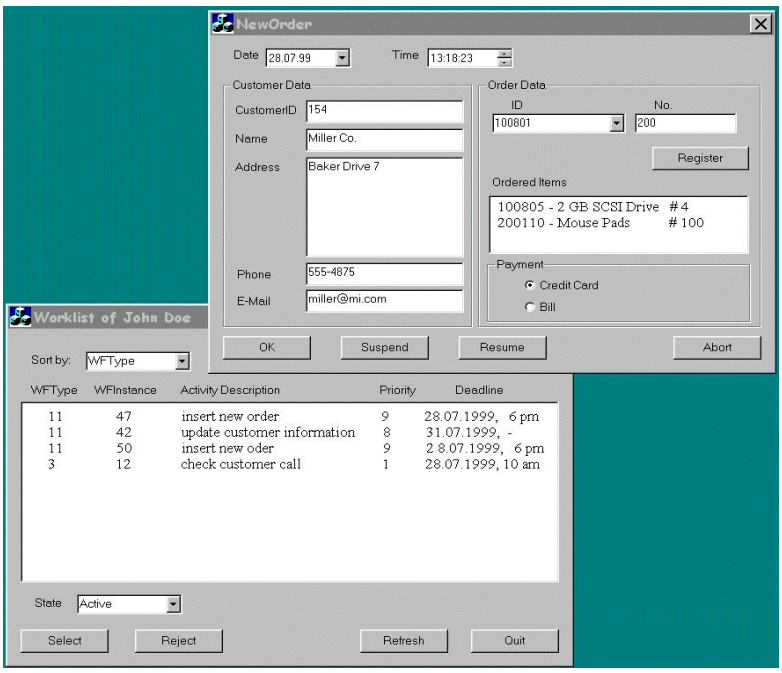

Figure 4 : Worklist of John Doe and user interface of activity "NewOrder"