Software Process Support over the Internet

F. Maurer & G. Succi
University of Calgary
2500 University Dr NW
Calgary, Alberta, T2N 1N4 Canada
maurer@cpsc.ucalgary.ca
Giancarlo.Succi@enel.ucalgary.ca

H. Holz, B. Kötting, S. Goldmann, B. Dellen
University of Kaiserslautern
AG Expert Systems
P.O. Box 3049
67653 Kaiserslautern, Germany
{holz,sigig,koetting,dellen}@informatik.uni-kl.de

ABSTRACT
The MILOS system supports software development processes over the Internet. It integrates process modeling with project planning and enactment. Our flexible workflow engine allows refining and changing process models during project execution. The built-in traceability component supports change notifications and helps the project participants to ensure that the project plan as well as the state of the enactment engine reflect the "real world" development process. Tool integration is accomplished by using the built-in capabilities of Web browsers.

Keywords
Process support, Internet, flexible workflow, change notification, traceability

1 INTRODUCTION

In most companies, software development still is more of an art than an engineering discipline. To overcome the resulting problems, companies are trying to improve their software processes following, for example, the capability maturity model (CMM) or similar approaches. Key issues here are process modeling and related activities: the development of explicit descriptions of how software has to be created and maintained. Most often, these descriptions are textual: Companies create process manuals and rely on their employees to interpret the contents of these documents while executing development processes.

To reduce the problems of textual descriptions (e.g. ambiguity, missing descriptions of some aspects of the software engineering process), various formal or semi-formal process modeling languages were developed (e.g. [1,3,8]). However, even (semi-)formal process models cannot ensure that the „real" development process follows the prescribed model; they still rely on humans to interpret and follow the model.

In order to solve this problem and to provide active guidance during process execution, enactment engines have been proposed (e.g. [6]). Basically, an enactment (or workflow) engine is an interpreter that operationalizes process descriptions and guides its human users in their software development tasks (thereby increasing the chances that the process model and the real-world process stay in sync).

Another main problem in software development projects is the dramatic shortage of skilled workers. It is often impossible to find appropriate people locally. This fact forces companies to create virtual teams (or even virtual enterprises) with members distributed all over the world.

In this paper, we describe MILOS, a Web-based process support system that improves the coordination and information exchange of virtual teams. Its flexible workflow engine allows the creation, refinement, and adaptation of project plans during enactment; therefore, it is suited for the highly dynamic, distributed environment of virtual teams. Its traceability and change notification mechanism supports team members in coping with changes.

In Section 2, we describe the system architecture. Section 3 explains the example for the demonstration whereas the last section discusses related approaches.

2 SYSTEM ARCHITECTURE

The MILOS environment consists of several components: a resource pool, a process modeling component, a project management component, and a workflow engine. These components are linked by a change management mechanism.

The *resource pool* component manages agents, roles and agent properties. It allows to represent company structures, e.g. in organization charts or hierarchical skill structures. These can be used to support scheduling of tasks by querying the component for agents that meet certain criteria.

The *process modeling* component maintains formal process definitions. These include control flow and data flow specifications, as well as pre- and postconditions, process refinement and required skills.

The *project plan management* component supports the project manager in planning and customizing the project. The manager can add dates for planned start and end times of processes and assign agents to processes. S/he can also change the project plan by adding or removing processes on every refinement level during execution of

ICSE '99 Los Angeles CA

the project. These operations can be performed using MS-Project.

The *workflow management* component is responsible for executing the project plan and managing products. It generates to-do lists for agents and maintains the current project state. The workflow engine is able to react dynamically to project plan changes during execution, without interrupting the enactment flow.

Our system supports different kinds of notification techniques. On one hand we provide standard notifications like escalation mechanisms (e.g. user notification on approaching deadlines). On the other hand, we generate notification dependencies from the project plan, and allow project participants to express interest in specific information. To implement these notification dependencies, we use Event-Condition-Action (ECA) rules that can be based on product and process-specific events like *product-changed* or *process-delayed.*

MILOS is implemented in Java. We are using the OODB GemStone/J 2.0 as an Enterprise Java Bean (EJB) server that provides transaction management and persistency services. EJB is a portable, highly scalable, multi-platform component architecture that dramatically simplifies the development of thin-client, multi-tier applications.

The technical architecture of MILOS is based on a multi-tier approach: many servers act as pillars (components) of the system, which is more in accordance with the OO paradigm than the traditional one-server-many-clients approach. We use three tiers: the process model, the project plan and the workflow.

The clients of our system are Web-based applets using Java Remote Method Invocation (RMI). Our replication and conflict solving mechanisms allow us to distribute workflow execution and also to execute parts of the workflow off-line.

As mentioned above, we implemented a connection to MS-Project for planning support. Other applications like MS-Office, FrameMaker and RationalRose can be invoked from within the execution framework.

For supporting the project and quality management, our system uses a metric tool developed at the University of Calgary by Succi[1] to measure different criteria during process enactment. The palette of measurable criteria starts at product specific measures like lines of code and goes up to process specific measures like effort.

3 EXAMPLE

In the following, we want to illustrate our system's functionality with the help of an example scenario describing our own project's software development process.

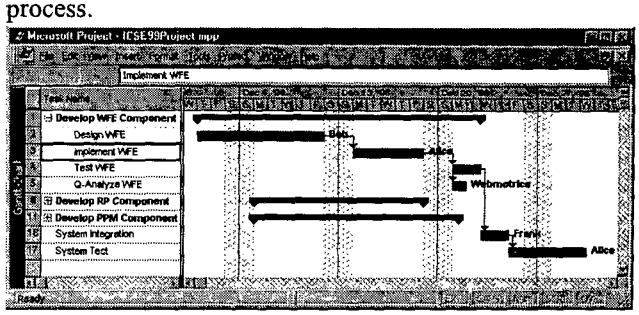

Fig. 1: Example plan created with MS-Project.

Based on the system's architecture, the project planner creates an initial plan using some standard software. The screenshot taken from MS-Project in Fig. 1 shows a simplified example plan for developing the MILOS architecture. For each of the three components *Project Plan Management* (PPM), *Workflow Engine* (WFE) and *Resource Pool* (RP), the plan contains a task[2] describing the component's development process. These tasks are complex, i.e. they are refined by a set of subtasks that describe the task (or the activities required to perform the task) in more detail. As Fig. 1 shows, the complex task *Develop WFE component* consists of the subtasks *Design WFE*, *Implement WFE*, *Test WFE* and *Q-Analyze WFE*[3]. Also shown is some scheduling information, i.e. planned start and finish times, duration as well as team members assigned to each task. In addition to the information shown in Fig. 1, the plan also contains a loop from *Test WFE* back to *Implement WFE*. This loop is modeled by specifying a product flow between those two processes, using MS-Project's additional task attribute fields.

For plan enactment, the project planner exports the plan from MS-Project into MILOS. From now on, team members, regardless of their geographical location, can log into MILOS via standard Web browsers and are provided with individual workspaces. Figure 2 shows the current workspace of team member Alice. According to the project plan, she is responsible for the task *Implement WFE* and, consequently, this task appears on her to-do list.

The workspace allows Alice to browse the information associated with each task, e.g. a more detailed task description (including the task hierarchy) and scheduling information. In particular, she is given access to any documents needed to execute the task. Hence, the list of input documents (the design document and the requirements document) as well as the list of output documents is displayed for task *Implement WFE*. These lists can be specified in the project plan. A double-click on a document opens it with the appropriate tool.

[1] http://www.sern.enel.ucalgary.ca/Charmi/ResearchProjects/Webmetrics/

[2] In the following, *task* and *process* are used synonymously.

[3] Quality analysis of code.

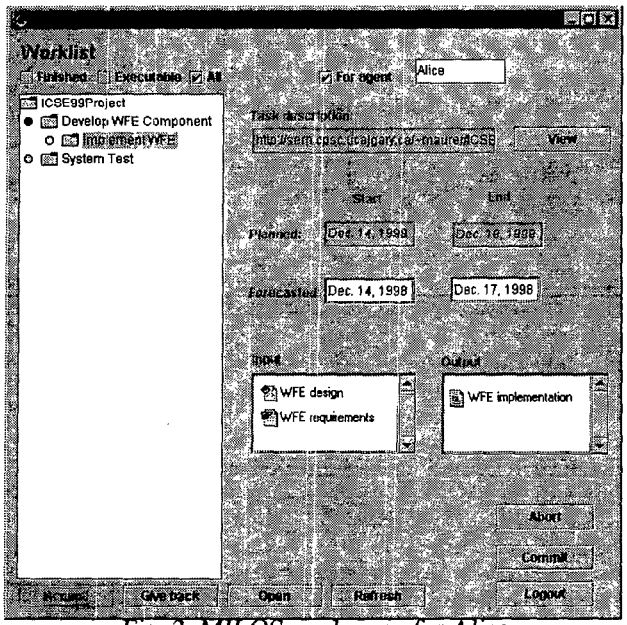

Fig. 2: MILOS workspace for Alice.

As soon as the required input document *WFE design* has been released by Bob, the team member who is responsible for task *Design WFE* (see Fig. 1), Alice is notified that the task *Implement WFE* has become executable. After having inspected the design document, she forecasts her start and finish times for the implementation to be the current date (Dec. 14) and Dec. 17, respectively. In the case that her forecast violated the project schedule, the planner would receive an automatically generated email notification about this problem. However, since her forecast conforms to the schedule, no notification is sent.

A double-click on the (empty) output document *WFE implementation* starts the appropriate Java implementation environment for Alice. At the end of each day that she is working on this task, she can save her work and specify a "percentage complete" value for it. This value will be exported from MILOS back to MS-Project in order to provide the planner with up-to-date information on the project.

When she is finished with the task, it will be removed from her to-do list. In addition, the document *WFE implementation* is released, to the effect that the two succeeding tasks *Test WFE* and *Q-Analyze WFE* become executable. According to the plan (see Fig. 1), the former has not been assigned to any team member yet, while the latter has been assigned to a specific metric tool. This tool acts as a software agent that performs the task automatically as soon as it becomes executable.

Depending on these measurements, the planner might want to refine the task *Test WFE* to either a black-box or white-box testing process. In our example, the planner settles for white-box testing because of a high number of conditional expressions in Alice's code. Hence, he refines the task *Test WFE* by creating two new subtasks *Write WFE Test Cases* and *Run WFE Test Cases*. In addition, he schedules these two new tasks and updates his former estimate on the finish time of the task *Test WFE*. Because this former estimate was based on his optimistic assumption that black-box testing would suffice, the planner now allocates more time for task *Test WFE*. As a consequence, the schedule for the succeeding tasks *System Integration* and *System Test* also has to be changed.

When he is finished with updating the project plan, the planner exports it again into MILOS. This causes the two newly created tasks to appear on the to-do lists of those team members the tasks were assigned to. In addition, the team members responsible for the tasks whose time scheduling had to be changed receive a corresponding notification. This allows them to update their own work schedule, in particular their forecasts on start and finish times.

In case any problems with Alice's code should be encountered during task *Run WFE Test Cases*, an MS-Word document containing a description of the problems will be created as output. The presence of this document will cause a restart of the task *Implement WFE*, i.e. it will appear once again on Alice's to-do list, together with the problem report as an additional input to the task. However, since by now the calendar has advanced to Dec. 28, whereas the implementation task was scheduled to finish by Dec. 18, the planner will receive an automatically generated email about this delay. That way, he will have the opportunity to correct the plan in time if project deadlines make this update necessary.

Meanwhile, Alice will release a new version of the document *WFE implementation* as soon as she has corrected the code. Analogous to the restart of task *Implement WFE*, the release of a new *WFE implementation* document version will cause a restart of the succeeding tasks *Test WFE* and *Q-Analyze WFE*. That way, a single restart might cause a "restart-cascade" that reaches all tasks affected by a change in a document.

4 RELATED WORK

Our work bears similarities to several areas of research, particularly project management tools, workflow management approaches, and process modeling and enactment research.

Commercially available project management tools like MS-Project[1] and Autoplan[2] support project planning and scheduling, but provide little or no enactment support. A project management system that does provide both planning and execution support is the Mesa/Vista Enterprise[3] tool. Mesa/Vista Enterprise is an environment for collaborative project execution and management. It provides distributed access to project data, as well as version and configuration management, but it does not include any change notification services.

Workflow management tools like Staffware[4], FlowMark[5], or TeamWARE[1] concentrate on project execution and

1 http://www.microsoft.com/project/

2 http://www.digit.com/

3 http://mesasys.com/vistapm/

4 http://www.staffware.com/

5 http://www.software.ibm.com/ad/flowmark

provide little or no support for process modeling and project planning. In particular, plan changes during enactment require a complete restart of the project in most workflow management tools.

The approaches most similar to our work can be found in the area of process modeling and enactment research. Most approaches in that area provide (web-based) modeling and enactment functionality, as well as some support for dynamic plan changes and change notifications. However, most of these approaches do not provide project planning and management support, like resource allocation and time scheduling for tasks in the project. Below, we briefly describe a number of approaches in the area of process modeling and enactment research.

Endeavors [2] is a support system for distributed execution of (workflow) processes. Endeavors provides support for dynamic process changes, and is currently being extended to support World Wide Web (WWW) protocols.

Serendipity [4] is a process modeling and enactment environment that supports collaborative modeling as well as execution of software processes. Change notifications are sent, using an event-handling concept similar to our approach. Several external tools have been integrated in the Serendipity system.

OzWeb [5] is a web-based system that supports multiple users who are grouped together into collaborative teams. OzWeb provides a framework that supports the storage of retrieval of information in a "referential hyperbase", and provides some notifications based on dependencies extracted from a process model.

EPOS [7] is a Software Engineering Environment with emphasis on Process Modeling, Software Configuration Management and support to cooperative work. The EPOS system is based on an underlying database, which provides versioning functionality and transaction management, controlled by an application-specific process model.

The SPADE [1] project aims at defining and developing a software engineering environment for software process modeling and enactment. Its process modeling language is based on a high-level Petri net formalism. The SPADE research also includes techniques to deal with process evolution during enactment.

STATE OF IMPLEMENTATION

The MILOS system is implemented in Java 1.1.7. The user interfaces run in any Java 1.1.7 compatible Web browser supporting Java Swing. The current versions of Netscape and Microsoft Internet Explorer need the Java Plug-in[2] installed. The Server side is implemented on top of the GemStone/J[3] 2.0.1 Enterprise Java Beans application server. The MILOS exporter/importer can be loaded into Microsoft Project 98. The tool is freely available for education and research purposes (contact maurer@cpsc.ucalgary.ca or koetting@informatik.uni-kl.de)

ACKNOWLEDGEMENTS

The work was supported by NSERC, Nortel, the University of Calgary, and the DFG with several research grants.

REFERENCES

1. S. Bandinelli, A. Fuggetta, S. Grigolli. Process Modeling-in-the-large with SLANG. In *IEEE Proceedings of the 2nd International Conference on the Software Process*, Berlin (Germany).
2. G.A. Bolcer and R. N. Taylor: Endeavors: A Process System Integration Infrastructure in *Proceedings of the Fourth International Conference on the Software Process*, Brighton, England, December 1996.
3. A. Bröckers, C. Lott, H. Rombach, M. Verlage: MVP–L language report version 2. *Technical Report 265/95*, Department of Computer Science, University of Kaiserslautern, Germany, 1995.
4. J.C. Grundy and J.G. Hosking,: Serendipity: integrated environment support for process modelling, enactment and work coordination, *Automated Software Engineering: Special Issue on Process Technology 5(1)*, January 1998, Kluwer Academic Publishers, pp. 27-60.
5. G. E. Kaiser, St. E. Dossick, W. Jiang, J. Jingshuang Yang and S. X. Ye,: WWW-based Collaboration Environments with Distributed Tool Services, *World Wide Web*, Baltzer Science Publishers (to appear).
6. G.E. Kaiser P.H. Feiler, S.S. Popovich: Intelligent Assistance for Software Development and Maintenance, *IEEE Software*, May 1988.
7. M.N. Nguyen, A.I. Wang, R. Conradi: Total Software Process Model Evolution In EPOS. *Submitted paper for 4th ICSP*, 1996, Brigthon, UK.
8. S. Sutton, L. Osterweil, D. Heimbigner: APPL/A: a language for software process programming, *IEEE Transactions on SE and Methodology*, Vol. 4, No. 3, p. 221-286, 1995

[1] http://www.teamware.com/

[2] http://www.javasoft.com/products/plugin/

[3] http://www.gemstone.com/products/j/main.html