Merging Project Planning and Web-Enabled Dynamic Workflow Technologies

FRANK MAURER
University of Calgary
BARBARA DELLEN
Fraunhofer Institute for Experimental Software Engineering
FAWSY BENDECK, SIGRID GOLDMANN, HARALD HOLZ, BORIS KÖTTING, AND MARTIN SCHAAF
University of Kaiserslautern

The MILOS system supports dynamic coordination of distributed software development teams by integrating project planning and workflow technologies over the Internet. The three-tiered Java architecture enables plan refinements to be made on the fly, and a change management component automatically creates traceability relationships between project entities.

One strategy for reducing time-to-market in software development is to deploy globally dispersed teams in concurrent work. This distributed development process requires flexible coordination and control to channel the work into a single, consistent system release. Several technologies have been developed to support distributed development. With project planning technology, managers can focus on scheduling and resource balancing issues using tools such as Microsoft's MS Project. These tools offer limited team support, however, because they cannot describe information flows. Nor do planning tools usually support coordination and routing of information to team members.

On the other hand, workflow systems and software process enactment (that is, execution or interpretation) engines are two technologies that do support process execution. Workflow systems, frequently used to automate business processes, route information between team members. Such systems are not readily adapted to the software development domain, however, because they usually lack planning support or do not support on-the-fly changes. Software process enactment engines provide more flexibility and a tighter integration with software engineering tools, but usually require specialized training in the modeling approach.

Yet another technology is project spaces, which apply Internet technologies to provide Web server access to project documents such as requirements, designs, and source code. Project spaces are passive, which forces software development teams to obtain information on their own behalf.

The active process support tool we describe in this article is MILOS, which stands for Minimally Invasive Long-term Organizational Support. MILOS is being developed as part of an ongoing joint project of the Software Process Support Group at the University of Calgary and the Artificial Intelligence Group at the University of Kaiserslautern. So far, the tool has been used in case studies for our own development process.

 1089-7801/00/$10.00 ©2000 IEEE

WORKFLOW SUPPORT CONCEPTS

Our approach is designed for highly *dynamic domains* in which the project tasks to be executed cannot be completely defined before the project starts. Also, the set of tasks in such a domain changes during project execution.

MILOS combines the active process guidance of workflow approaches with the flexibility and information access of project spaces. MILOS thus supports software development team members by providing to-do lists that let developers access task-related documents and background information; by automating information flow between team members; and by automatically launching the right tools for individual tasks.

MILOS gives project planners the flexibility to make on-the-fly changes. First, its workflow engine allows modifications to the process definition while executing it; the modifications are initiated by the plan changes made by the project's planners. Second, and more important, the system has a built-in change management component to handle the effects of process changes, which prevents project inconsistencies that easily occur when team members are geographically dispersed.

Instead of managers' having to learn a process modeling language, MILOS follows an evolutionary approach by extending Microsoft's MS Project with the means to describe information flow, as we explain later. A team can adapt the level of active process support to the project's current needs. The support can start from basic to-do lists, expand to task descriptions enriched with project-document

Related Work

Our work relates to research in software process modeling approaches, workflow management systems, and traditional project management approaches. For example, EPOS[1] and SPADE[2] provide process evolution support. In contrast to MILOS, these systems do not support change notifications very well, so system users cannot express interest in certain changes.

OzWeb[3] focuses on the Web as an artifact repository. It uses a rule-based process modeling language and a process engine featuring forward and backward chaining. OzWeb has no notification support for planners.

Serendipity[4] lets team members express interest in certain changes in the form of event-condition-action rules or *filters*. Unfortunately, change rules can be specified only on the instance level; furthermore, the approach does not discuss how to handle project plan changes.

Endeavors, an open, distributed, extensible process execution environment,[5] is designed to improve coordination and managerial control of development teams. Endeavors supports dynamic process changes over configurable enactment models and an event monitoring structure and integrated support for communication between participants. Nevertheless, it provides only weak support for notification and replanning.

Workflow management systems have been successfully used in business applications (for general information on the industry, see the Workflow Management Coalition home page at http://www.aiim.org/wfmc/). Staffware (http://www.staffware.com), Teamware Flow (http://teamware.fujitsu.com.au/teamware/Products/Process/flow.htm), and FlowMark (http://www.software.ibm.com/ad/flowmark)—large vendors with systems having features somewhat similar to those of the MILOS system—are not integrated with project planning tools, however. They lack the flexibility to interleave planning and execution, and support only a weak form of on-the-fly process changes that has not been well integrated with change management and notification facilities.

Project management support systems integrated with the Internet include Microsoft's MS Project, MesaVista (http://www.mesasys.com), Platinum Process Continuum (http://www.platinum.com/products/appdev/ppcpr.htm), and iTeamWork (http://www.iteamwork.com). Although specific subsets of the functionality that MILOS offers can be found in MesaVista or Process Continuum, a main advantage of our approach lies in the cohesive framework achieved by technique integration.

The Process Link project (http://cdr.stanford.edu/ProcessLink) at Stanford University follows a dependency management approach similar to MILOS: The Redux design model[6] states that goals, for which decisions can be made, guide the artifact design activity. Between those decisions, dependencies can be stated that enable Redux to automatically notify users of changes. Furthermore, Redux can achieve pareto optimality by notifying the user when constraints are relaxed, and can reject a suboptimal solution in favor of a better one.

In the Procura approach,[7,8] the model supports concurrent design, planning, and plan enactment in highly dynamic domains such as mechanical and civil engineering applications. MILOS, on the other hand, adds specific process and product models to increase the number of automatically generated dependencies.

references, and culminate in full-fledged document routing from producers to consumers, including tailored change notifications.

The integration of project planning and workflow support gives software development teams an easy improvement path from current practice (a loose coupling between planning and execution) to more active support (generating to-do lists, updating them, and making notifications in case of deviations or inconsistencies). MILOS achieves this integration by supporting MS Project as a planning interface for our flexible workflow engine.

Similarly, software developers prefer to use their own favorite tools to edit their products. When a user selects a requirements document to work on, MILOS will, for example, automatically launch Microsoft's MS Word. When a user selects Java code to work on, MILOS will automatically launch IBM's VisualAge for Java.

To implement these concepts in a distributed environment, users download appropriate plans and products from the server over the Internet and upload them after task completion. Remote access is transparent to users.

Process Models and Project Plans

To provide tool support for software development processes, MILOS requires explicit process definition as the process evolves. To define processes generally, a modeling language specifies different facets of software development such as processes, products and product flow, and resources.

Based on a model of a company's general software development process, a project can be planned. (Planners can create plans from scratch, but they might find it convenient to choose already defined task sequences, including information flows, from a process model library.) While a process model represents development processes and contains neither timing nor resource allocation information, a project plan must specify exactly who must do what and when. We chose MS Project because most software project managers are familiar with it. However, MS Project does not support information flow, a concept essential to enactment support in which document routing and information-flow-based notifications are key. We therefore extended MS Project to support these features via definition of task input and output parameters. MILOS represents information flow between processes by mappings: There exists a product flow between tasks A and B if one of A's output parameters is mapped to one of B's input parameters.

Besides product flow, several other extensions in the MILOS system do not yet exist in MS Project but are accessible via the MILOS user interface. The most important concepts follow.

- *Alternative solution methods:* For one process, MILOS lets planners specify several possible decomposition methods in the process model. The planner can decide which alternative method to use without creating a new process subplan.
- *Process preconditions and postconditions:* These specify what conditions must hold before the process starts or after the process completes. They can specify control flow or define quality conditions on process results.
- *Product model:* MILOS includes several predefined product types, each with an associated file type (for example, Rational Software's Rational

References

1. M. Letizia and R. Conradi, "Techniques for Process Model Evolution in EPOS," *IEEE Trans. Software Eng.*, Vol. 19, No. 12, Dec. 1998, pp. 1145-1156.
2. S. Bandinelli, E. Di Nitto, and A. Fugetta, "Supporting Cooperation in the SPADE-1 Environment," *IEEE Trans. Software Eng.*, Vol. 22, No. 12, Dec. 1996, pp. 841-865.
3. G.E. Kaiser et al., "WWW-based Collaboration Environments with Distributed Tool Services," *World Wide Web J.*, Vol. 1, No. 1, 1998, pp. 3-25.
4. J.C. Grundy and J.G. Hosking, "Serendipidity: Integrated Environment Support for Process Modeling, Enactment and Work Coordination," *Automated Software Eng.*, special issue on process technology, Vol. 5, No. 1, Jan. 1998, pp. 27-60.
5. G.A.Bolcer and R.N. Taylor, "Endeavors: A Process System Integration Infrastructure," *Proc. IEEE Computer Soc. Int'l Conf. Software Process* (ICSP4), IEEE Computer Soc. Press, Los Alamitos, Calif., 1996, pp. 76-89.
6. C. Petrie, "The Redux Server," *Proc. Int'l Conf. Intelligent and Cooperative Information Systems* (ICICIS), IEEE Computer Soc. Press, Los Alamitos, Calif., 1993, pp. 134-143.
7. S. Goldmann, "Procura: A Project Management Model of Concurrent Planning and Design," *Proc. WETICE-96*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1996, pp. 177-183.
8. C. Petrie, S. Goldmann, and A. Raquet, "Agent-Based Project Management," *Lecture Notes in AI–1600*, Springer Verlag, Berlin, 1999.

Rose visual modeling tool and Microsoft Word text editing software). This enables the enactment engine to launch the appropriate tool when the user selects a development product.
- *Agent model:* Agents (development team members) in MILOS have properties and can play specific roles. In the process model, roles and property requirements describe the skills needed by an agent to execute a specific process.

MILOS's event mechanism notifies team members of changes and informs developers of their projects' progress.

A MILOS process model implicitly specifies dependencies between processes and products. The workflow engine makes them explicit for change management and change notifications.

Workflow Engine

The MILOS workflow engine stores project state information, including deliverables created during enactment. It routes new versions of deliverables to the appropriate team members, and generates to-do lists that provide personalized project views to every team member. For every process in the to-do list, the workflow engine offers access to background information, such as the process description, guidelines, and projected start and end dates.

In MILOS, all changes generate events: Workflow engine objects react to changes in project plan objects. When the project plan is updated, the MILOS system determines the effects and responds by implementing the corresponding adaptations of the workflow engine's state. Plan updates recognized and handled by the workflow engine include schedule changes, process addition and removal, resource assignment changes, and product flow changes.

Besides using the event mechanism to update the workflow engine's internal state, MILOS uses this mechanism to send e-mail notifications to team members whose work the change affects. Similarly, the workflow engine uses this mechanism to inform developers of progress, such as when required project deliverables become available.

The ability to manage changes in both project plan and project state is based on change rules. A default set of rules lets the workflow engine specify how to update the state after change occurs. In MILOS, change rules are the core mechanism for handling software project dynamics. The next subsection gives more insight into the underlying techniques.

Change Notification Support

Active change support requires the management of project dependencies. Project dependencies, if known, can be automatically analyzed to determine the effects of a change.

We use a rule-based approach to represent knowledge about effects of changes. Event-Condition-Action (ECA) rules allow us to specify the origin and effect of a change. We distinguish between two kinds of rules:

- rules that enforce the automatic update of affected process and product information, and
- rules that specify who should be informed about a change.

Typically, a change in the project plan, product updates, or state changes of processes causes change rules to fire. Team members can tell the MILOS system on which changes they want to be notified by specifying change rules.

Unfortunately, eliciting this information from human users is a massive effort, which MILOS can reduce via the specification of change rules that apply to more than one case. We call these generalized change rules *change rule patterns*. Once established, the patterns can be instantiated several times during project enactment. Suppose, for example, that two processes A and B are related by a product flow relationship, so that the outcome of A is a prerequisite to conduct process B. A change rule pattern states that whenever A's outcome is updated, the person in charge of B must be notified.

The default set of change rule patterns in MILOS deals, for example, with product flow relationships, schedule inconsistencies, and product updates. The workflow engine uses these patterns to automatically generate causal relationships among project entities, reducing the user's burden for manually defining change dependencies.

Beyond these generic patterns, MILOS features domain-specific change rule patterns, which differ from the domain-independent, generic change rule patterns in how they apply knowledge about pro-

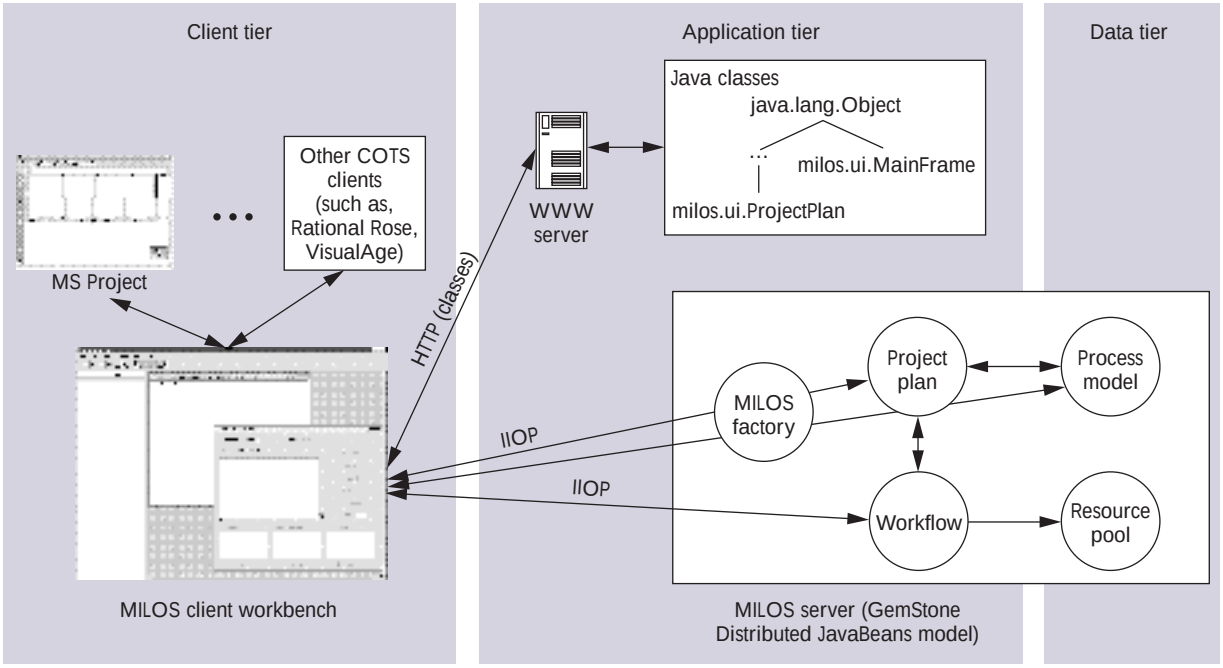


Figure 1. The MILOS system is based on a three-tiered architecture. The graphical user interface accesses an object-oriented database management system that implements the business logic and provides data access.

ject or domain characteristics. An example is the relation between Unified Modeling Language class specifications and the corresponding code. With a UML-based design, the names of all Java classes correspond to the related UML classes. Based on this observation, a domain-specific change rule pattern could enforce notification of the developer of a Java class whenever the related UML class specification changes.

SYSTEM ARCHITECTURE AND IMPLEMENTATION

Figure 1 shows the MILOS system architecture. The MILOS factory is responsible for instantiating the MILOS objects and creating remote references to them that will be distributed with the underlying CORBA communication infrastructure.

Conceptually, the architecture takes a three-tiered approach. On the client tier, the MILOS workbench provides graphical user interfaces for planning and enactment, and is responsible for launching appropriate tools for documents.

The application tier consists of the application logic and is embedded into GemStone Systems' GemStone object-oriented database management system that also covers the data tier. GemStone provides typical services for both tiers, such as program execution, persistency, security, and transaction management. Consequently, neither an additional relational database nor a dedicated application server is required. The OODBMS also offers an object-oriented view on persistent data, and, with GemStone's proprietary Distributed JavaBeans application model, transaction boundaries can be different from method boundaries. Additionally, GemStone offers a convenient migration path toward the Enterprise JavaBeans application model, a standard framework for accessing and realizing object persistence on top of conventional relational database management systems.

The MILOS graphical user interfaces can be dynamically loaded via a Web server and executed as (signed) applets within a Web browser or as standalone applications. Communication with server-side components (for example, a project plan) is done via CORBA IIOP, which will permit communication with non-Java clients if required in the future.

EXAMPLE SCENARIO

The following scenario illustrates how the infrastructure provided by MILOS supports software development projects. To set up a project plan, a planner logs into the MILOS server using a Java applet. This starts a workbench applet, which provides access to company-specific knowledge

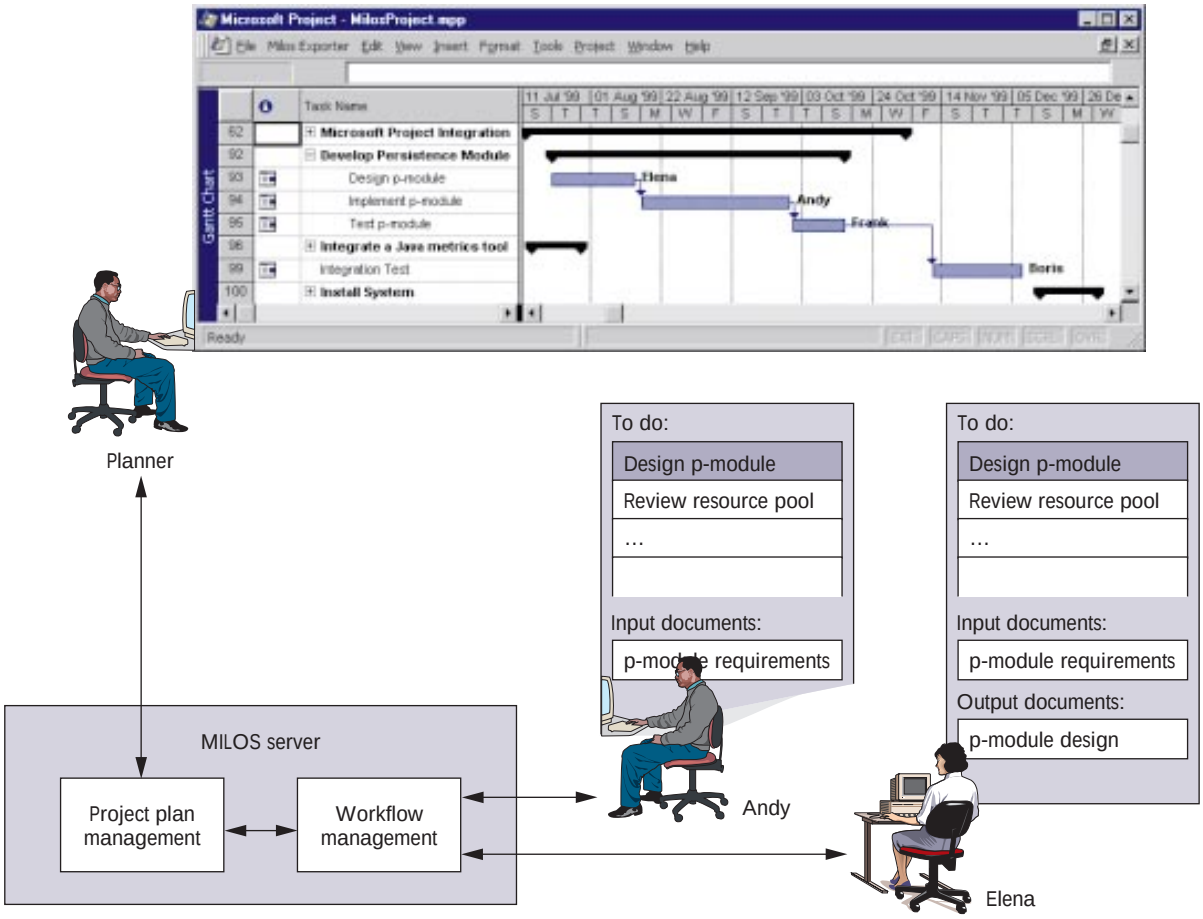


Figure 2. A software development project planner uses MS Project to create an initial project plan. The plan is then transferred to the MILOS system, which, in turn, updates the to-do lists of all team members.

resources (especially a process model library and a resource pool containing the skill model).

Project Planning

To create an initial plan or change an existing plan, the workbench applet launches the project planning tool. Figure 2 shows part of a project plan as created during a software development process. The figure depicts the plan at an early stage, in which task Develop Persistence Module consisted of only three subtasks: Design p-module, Implement p-module, and Test p-module. When plan editing is complete, the planner uploads the plan to the MILOS server and automatically imports it, starting enactment.

To tailor the default information flow between tasks provided by the process model, the planner uses the MS Project extensions described earlier. For example, task Design p-module has the input parameter "p-module requirements" of type ComponentRequirements and an output parameter "p-module design" of type ComponentDesign. Figure 3 depicts these dependencies as an activity diagram.

The Implement p-module task has an input parameter "p-module design," which is mapped from the design task's output and used as an input to the implementation task.

While the plan already contains enough information to begin enactment, the planner decides to constrain the order of task execution by specifying task preconditions. For example, the planner sets the precondition of task Integration Test to the predicate "p-module.errorRate < 10," stating that integration testing is not allowed to begin until the p-module's error rate is less than 10 percent (see Figure 3).

Plan Enactment

Once plan enactment begins, team members are notified automatically by e-mail about tasks assigned to them. Team members can log into the

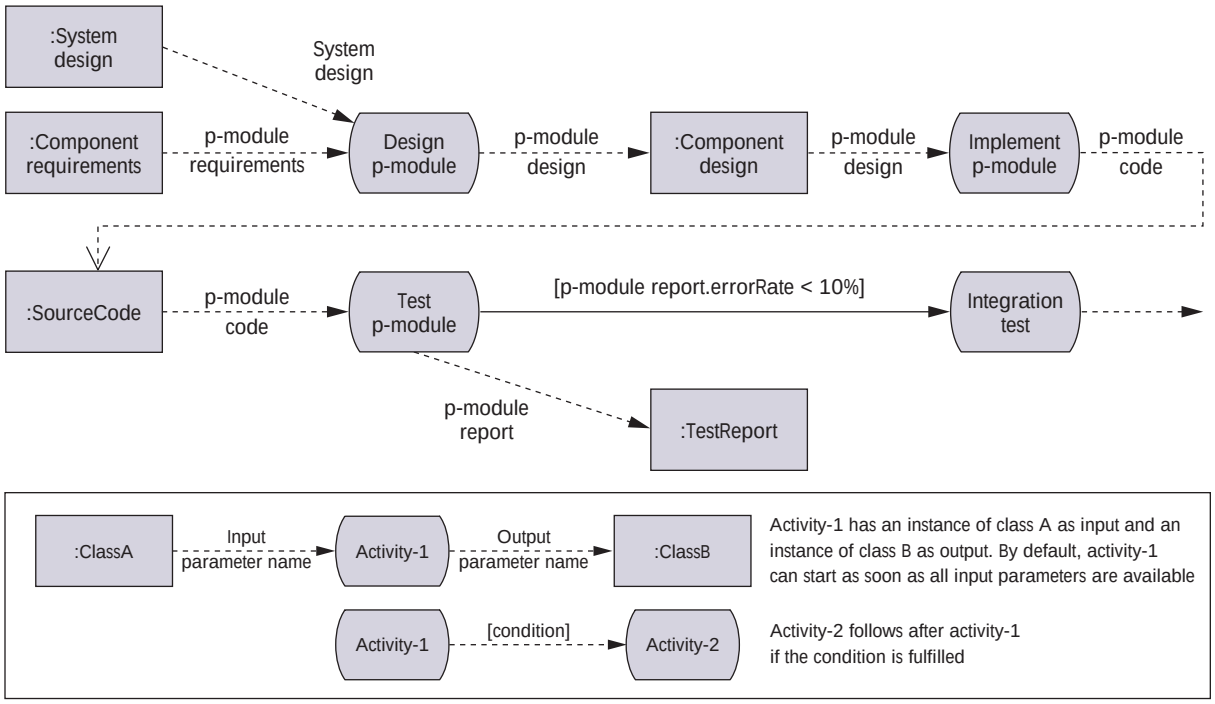


Figure 3. Activity diagram represents the information and control flow between tasks, which a planner can use to augment an initial MS Project plan.

MILOS server via applets and be given access to relevant information, such as their to-do lists and process states. Figure 4 shows a screenshot of a project member's workspace and a subsequent plan update. Since the task Design p-module has been assigned according to the plan, this task appears on the to-do list. Additional task information—specifically, the documents referred to by the task's input and output parameters—can be accessed from the workspace's lower area.

The document assigned to the input parameter "p-module requirements" is the output of a preceding task in the plan. As soon as the project member responsible for this task releases a version of this requirement document, Elena will automatically receive e-mail notifying her of a (new) input document for the task Design p-module. Selecting the parameter "p-module requirements" will cause the current (released) version of the requirement document (the result of a preceding task) to be downloaded to Elena's workstation and, according to the parameter's type information, opened with MS Word.

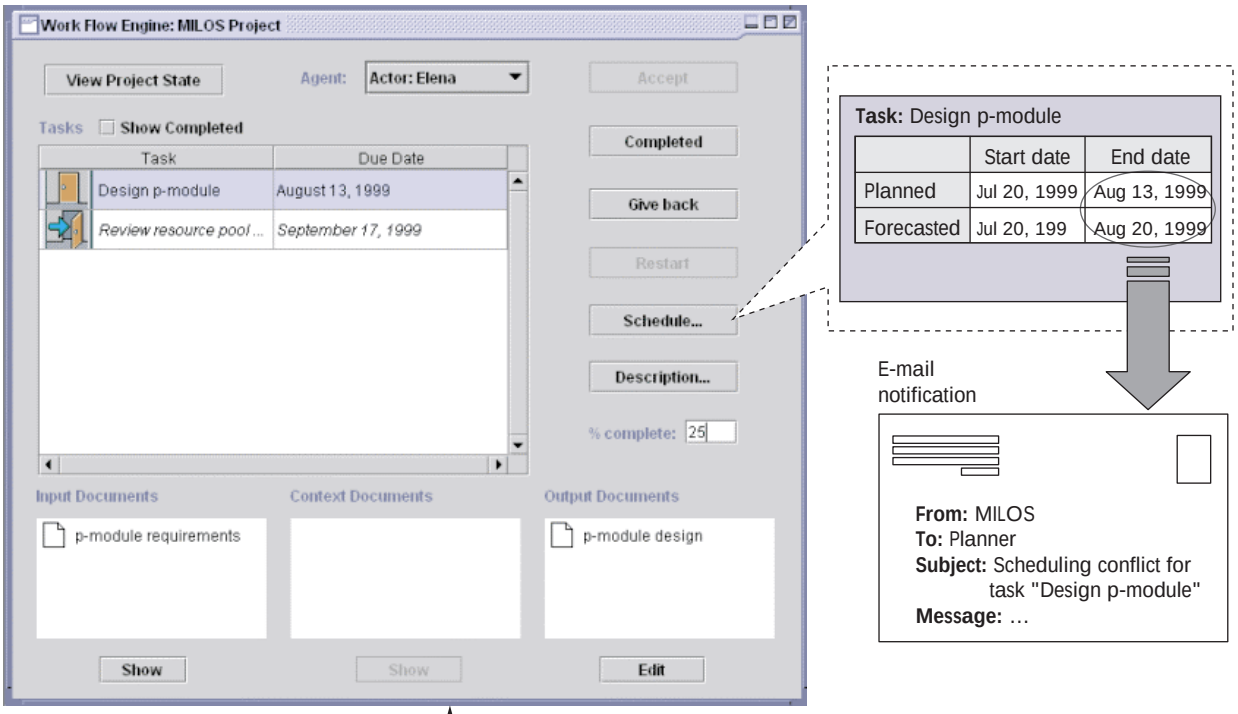
By studying the module requirements document, Elena can more realistically estimate the time she needs for the design than the planner could estimate before the requirements analysis was done. She thus accesses the task's detailed scheduling information and enters her forecast (see the circle in the upper right of Figure 4). Since her forecast is inconsistent with the project schedule, the planner receives an automatic e-mail notification concerning this problem. Consequently, the planner contacts Elena to discuss it and updates the schedule accordingly.

Elena starts working on her task by selecting the output parameter "p-module design," which, according to the parameter's type (not visible in the figure), launches Rational Rose with an empty document. At any time, she can notify the planner of her progress by specifying a percentage-complete value. Information on project progress is available both in MS Project and via a project state view provided by MILOS.

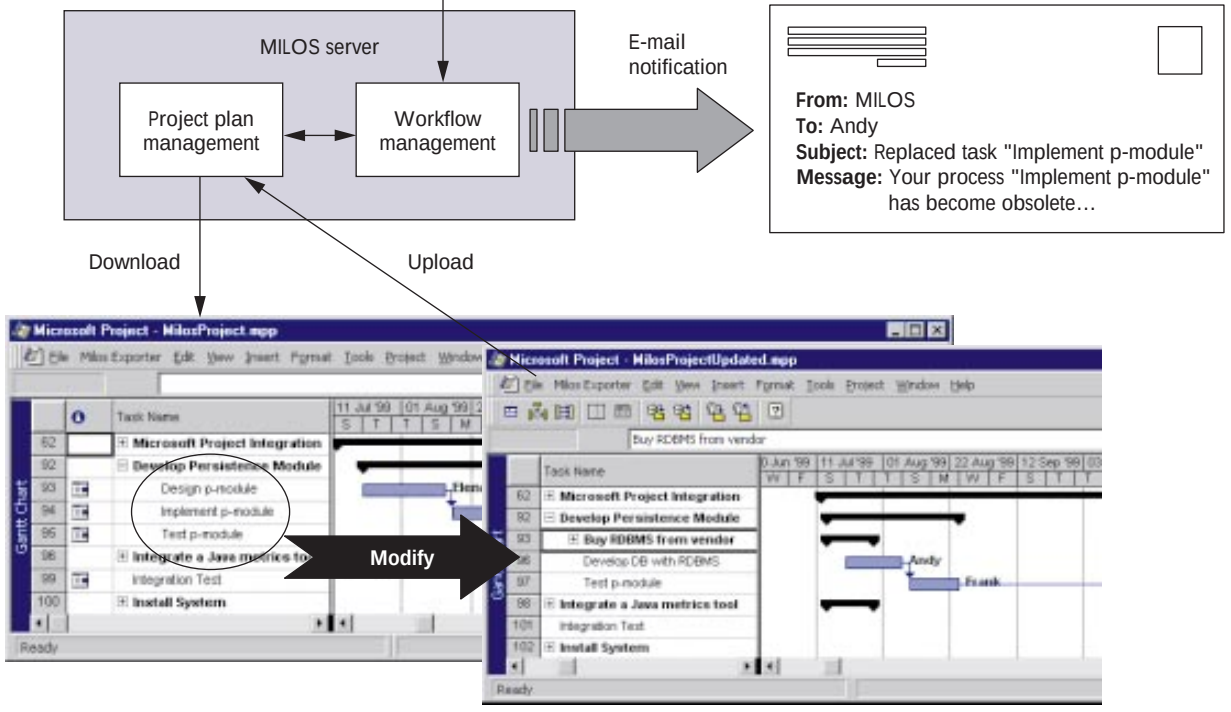
A dialog window for output parameters lets Elena "freeze" her work as a version and release it as the current product for the corresponding output parameter. This causes MILOS to upload the document to the server and notify Andy, responsible for the succeeding task to implement the persistence module, of the input document's availability.

A Plan Change

What if a change in the system requirements forces the planner to update the development process concerning the persistence module? Instead of the



(a)



(b)

Figure 4. (a) The MILOS workspace screenshot. Entering a forecast that violates the planner's scheduling results in an e-mail notification from the MILOS system to the planner. (b) A subsequent plan update renders team member Elena's design task obsolete, and team member Andy's implementation task has been replaced by task Develop DB with RDBMS. Frank, a team member listed in the project plan at the bottom, has had his task rescheduled accordingly.

file-based solution we are developing in-house, the persistence module must be based on a commercial off-the-shelf RDBMS. To change the plan, the planner logs into MILOS and re-opens the current project plan. Since purchasing a commercial product and building an implementation on it is a standard software development process, our MILOS process library already contains a corresponding process model.

The planner downloads a process model from the process library and integrates it into the plan (for example, by changing the process name "Buy product from vendor" into the more specific task name "Buy RDBMS from vendor," creating parameter mappings for "dangling" parameters and so on). The updated plan is depicted at the lower right of Figure 4. When the planner uploads it to the MILOS server, all team members affected by the plan changes receive automatic e-mail notifications. For example, Andy will be notified that he can stop working on the current implementation and instead start developing a database. The notification is caused by the following ECA rule:

```
EVENT: REPLACE(p:process, q:process)
CONDITION: -
ACTION:
NOTIFY(owner(p), "Your process", p, "has
become obsolete. Please save any results already
created and stop working on it.");
NOTIFY(owner(q), "A new process", q, "has been
assigned to you.")
```

The variables p and q are replaced by concrete values during enactment; the term owner(p) determines the person that is currently in charge of process p. Also, since the development based on the commercial product is assumed to be finished earlier, the task schedule has changed. Consequently, the team member responsible for task Test p-module will be notified that the task has been re-scheduled for an earlier start date. This notification is initiated by an ECA rule of the form

```
EVENT: REPLACE(startDate(s: time, p:process),
startDate(t: time, p:process)))
CONDITION: s > t
ACTION: NOTIFY(owner(p), "The start time of your
process has been rescheduled to an earlier time")
```

Backtracking to an Earlier Version

Although Andy works on the new task, Develop DB with RDBMS, and completes the table definitions, he cannot start the database implementation because the DBMS software still has not arrived. He notifies the planner of this problem. Because a fixed milestone states that a first version of the system, including persistence functionality, must be up and running in one month, the planner again changes the plan: He re-inserts Andy's task Implement p-module from the former plan, telling him to resume work on the p-module implementation as long as the RDBMS doesn't arrive. Because of the task-oriented version management MILOS provides, Andy can easily resume his work, again being given direct access to Elena's design document as well as to his formerly created products associated with the reactivated task. Similarly, Frank can return to a corresponding version of his test cases.

An enhancement to MILOS is support for virtual corporations—a temporary network of suppliers and customers.

FUTURE WORK

We are currently extending the MILOS environment by a process-centered experience base to support managers in finding team members with a specific skill set and proposing methods for tasks.

We will migrate the MILOS system from the GemStone-proprietary Distributed JavaBeans model to the standard Enterprise JavaBeans component model. Once finished, a team needs only an EJB-compliant application server (for example, IBM's WebSphere; Oracle; or GemStone) to run the MILOS server.

Another enhancement will be support for *virtual corporations*—a temporary network of independent companies, both suppliers and customers—linked by information technology to share skills, cost, and access to one another's markets. Our approach will support the subcontract negotiation process and the distribution of subprojects to several workflow engines.

The MILOS system is planned for summer 2000 availability, under an open-source license via the MILOS Web sites (http://sern.ucalgary.ca/~maurer/Research/research.htm and http://wwwagr.informatik.uni-kl.de/~milos/).

ACKNOWLEDGMENTS

The MILOS project is supported by Nortel, the National Science and Engineering Research Council of Canada NSERC, the Alberta Software Engineering Research Consortium ASERC, the Deutsche Forschungsgemeinschaft DFG, the University of Calgary, and the University of Kaiserslautern.

Frank Maurer heads the software process support group at the University of Calgary and is codirector of the Alberta Software Engineering Research Consortium. His research interests include Internet technologies for software engineering, experience management, and dynamic workflow. Maurer received a PhD in computer science from the University of Kaiserslautern.

Barbara Dellen heads the process engineering group at the Fraunhofer Institute for Experimental Software Engineering, Germany. Her research focuses on change impact analysis in software processes and workflow management technology for software process improvement. She received a PhD in computer science from the University of Kaiserslautern.

Fawsy Bendeck is working on a PhD at the University of Kaiserslautern. His research interests include software engineering and abstraction from an experience base by means of evolutionary programming. Bendeck received a BS in computer science from the Jose Cecilio del Valle University, Honduras, and an MSc from the Technological Institute of Costa Rica.

Sigrid Goldmann is working on a PhD at the University of Kaiserslautern. Her research interests include metaplanning support in workflow management systems and software engineering. Goldmann received the Diplom from the University of Kaiserslautern.

Harald Holz is a researcher at the University of Kaiserslautern. His research interests include the integration of project support and knowledge management systems, case-based reasoning, and design patterns.

Boris Kötting is working on a PhD at the University of Kaiserslautern. His research interests include distributed workflow management systems and distributed artificial intelligence. Kötting received the Diplom from the University of Kaiserslautern.

Martin Schaaf is working on a PhD at the University of Kaiserslautern. His research interests include knowledge management techniques and distributed systems. Schaaf received the Diplom from the University of Kaiserslautern.

Contact the authors at maurer@cpsc.ucalgary.ca; Barbara.Dellen@iese.fhg.de; or {bendeck, sigig, holz, koetting, schaaf}@informatik.uni-kl.de.