Integrating Process Support and Knowledge Management for Virtual Software Development Teams

Frank Maurer

University of Calgary

Department of Computer Science,

Calgary, Alberta, Canada, T2N 1N4

maurer@cpsc.ucalgary.ca

Harald Holz

University of Kaiserslautern

Department of Computer Science,

D-67653 Kaiserslautern, Germany

Abstract

In this paper we describe how knowledge management and software process support can be integrated to improve the efficiency of virtual software teams. The approach presented here integrates a process enactment environment with an on-demand knowledge delivery strategy that is based on parameterized information needs models. The parameters in the information needs models are bound at project execution time to values extracted from the process enactment engine. Thus, the approach supports virtual teams by establishing a platform for systematic and task-specific knowledge exchange. The proposed approach is prototypically implemented in the MILOS system, an open source project of the University of Calgary (Canada) and the University of Kaiserslautern (Germany).

1. Introduction

In the 1990's, virtual software development organizations were emerging: teams of developers that work on the same software but are distributed all over the world began using telecommunication and the Internet for communication, collaboration and coordination. Some examples are software development projects in the telecom or military industry where teams are distributed over the whole country or even the whole world. Other examples are open source projects like Linux[1] or the Apache Web Server[2]. These projects delivered complex software of high quality but the software development team never or, at least, very rarely met in the real world.

In the same decade, freelance work dramatically increased [Laubacher, Malone 1997], especially in the high tech field. As a result, Thomas Malone and his co-workers described the emergence of an e-lance economy [Malone, Laubacher 1998] where freelance workers collaborate virtually over electronic networks to perform their jobs. E-lancing works best when all job related information and work results can be transferred over electronic networks. Software development is, hence, a prototypical example for this kind of work organization. This is demonstrated by new ventures like eLance[3] or Asynchrony[4] that are building virtual marketplaces for e-lance work and whose focus is on software development and web design.

The business advantage of virtual teams and e-lancing are primarily twofold:

- Increased flexibility in finding required resources when they are needed
- Reduced costs because of outsourcing to markets providing cheaper labor and less training expenses

The trade-off consists of human costs. Social security for e-lance workers is usually less then for employees. As a result, Laubacher and Malone anticipate the emergence of "New worker's guilds" [Laubacher, Malone 1997]. These guilds would provide benefits like health care, pension and unemployment insurance. In addition, they would act as social networks – in a sense, replacing some side effects of employment.

A core aspect of the new guilds will be to guarantee a minimum level of expertise of their members and provide training and education as part of the continual learning processes required in the knowledge-based economy. In a sense, these learning processes of the new guilds will complement the learning processes of corporations. The question addressed in this paper is now:

---

[1] http://www.linux.org
[2] http://httpd.apache.org
[3] http://www.elance.com

How can we build tool support that integrates process execution and continual learning for virtual software teams?

We first discuss how virtual software teams currently are working using open source projects as examples. Then we provide an overview on the MILOS approach illustrating how it integrates process support and continual learning. The focus of this paper - the knowledge management aspect of MILOS – is discussed in Section 4. Section 5 explains how continual learning is supported by MILOS. We then describe the state of implementation and related work. The last section summarizes the paper and glimpses at future work.

2. Virtual Software Teams – State of Practice

Open source projects are good examples of software development where self-motivated individuals work together in a virtual environment to produce software. Analyzing their processes lets us determine

- How virtual software teams currently communicate, collaborate and coordinate their work
- How they share information and knowledge
- What tool support they use
- Where shortcomings are

Besides violating most standard software engineering practices regarding documentation of requirements and design, most open source projects demonstrate some common approaches how the development process is organized and how knowledge is transferred between the heads of team members.

2.1. Organization of Open Source Development Processes

Open source projects need to ensure that relevant information is available to all participating developers. Here we need to differentiate between which information is seen as relevant (contents) and how it is made available to the developer community (modus).

All well-known open source projects maintain one or more Web site(s) as a focal point for information exchange. The Web site usually describes the mission and goal of the project and provides access to project related information.

Source code and documentation is usually accessible as one compressed file or via an Internet-based version management system (most often CVS [CVS 2001] with a web-based front end).

---

[4] http://www.asynchrony.com/welcome.jsp

Issue tracking systems, using a Web-based front end, are used to record bugs and manage the workflow for fixing them. Usually, there are clear definitions when to submit a new bug report and how it will be handled.

Discussion groups or mailing lists are used to propose new features and extensions[5]. Newsgroups and mailing lists can also be used for getting help in case of problems with the open source software.

Most information is made available to the developer community in pull mode: developers are able to access the Web site of the project including the source code tree whenever it fits into their schedule. The same holds for discussion groups. This gives the developers control of their time. Push mode is e-mail based. It is used for distributing bug reports, for discussing new features and for on-line support. Although e-mail is more intrusive than newsgroups, it still is an asynchronous communication medium.

Overall, open source development processes primarily use Internet technologies for supporting communication and collaboration. For example, the Apache Group states: "using the Internet and the Web to communicate, plan, and develop the server and its related documentation" [Apache 1999]

An important aspect of open source development is providing recognition of individuals. Mailing lists and discussion groups show names or "screen names" of developers. This provides a sense of community to the developer group – in a sense, this is fulfilling some aspects of a software guild. In addition, the set of core developers is usually acknowledged on the project's web site. In fact, core developers have higher privileges than others concerning changes to the source code and voting about the contents of next releases.

One question that is still unanswered is if the open source approach is effective and if it can be transferred to a business environment. Clearly, some open source processes are very efficient as shown by very short times – sometimes measured in hours - for fixing bugs that were posted to the bug tracking system. But there is currently no study that discusses how much development effort is wasted on code that never makes it into a release of the software.

2.2. Knowledge Management in Open Source Projects

Knowledge management is very basic in open source projects. Knowledge is shared via the Web site. Sometimes a basic skill level is defined and documents are provided to reach this skill level. For example, the Jakarta project states:

"A common foundation of knowledge is required to effectively participate in this virtual community." [Apache 2001]

Knowledge is often preserved and condensed in the form of frequently asked questions (FAQs). Process knowledge, meaning knowledge about how the development process is carried out by the group, is only informally documented – if it is documented at all - in form of web pages.

Besides browsing, some open source web sites provide standard text retrieval facilities that allow searching for the appearance of text strings in the content of the site.

In addition to using the web site to acquire knowledge about the project, knowledge is exchanged via mailing lists and newsgroups. These basic knowledge management facilities make it very difficult for project newcomers to find all relevant, available information for a development task that she wants to undertake.

3. The MILOS Approach – An Overview

The overall goal of the MILOS[6] approach is to support process execution and organizational learning for virtual software development teams; this support should be minimally intrusive to reduce overhead. The MILOS approach can be applied in open source projects as well as in commercial teams that are distributed over the world. Using its knowledge management features, it can also serve as a tool for software guilds.

3.1. Process Execution Support in MILOS

To provide the background required to understand the knowledge management extensions of MILOS, we here give a short overview on the support offered by MILOS to virtual teams for process execution. More details can be found in [Maurer et al 2000]. The next section explains how MILOS integrates process support and knowledge management for virtual teams.

MILOS supports the execution of globally distributed software development projects in several ways:

Project coordination: MILOS provides standard project coordination support for virtual teams. It allows assigning tasks to developers, setting deadlines and getting an overview on the current state of the project. MILOS manages traceability relationships between process entities [Dellen et al 1996]. Team members are able to access their to-do lists and retrieve relevant information for performing their tasks easily.

Document routing and active notifications: Using a standard Web browser, a user is able to connect to the workflow engine[7], access her to-do list, accept/reject/restart a task, access task inputs and background

---

[5] Sometimes new feature requests are stored in the issue tracking system.

[6] **M**inimally **I**nvasive **L**ong-term **O**rganizational **S**upport.

[7] The workflow engine is also called “process engine” or “process enactment engine”.

information, and edit task outputs. After completing a task, the outputs are accessible as input for successor tasks – realizing document routing between distributed team members. MILOS uses a version control system to manage all files that are produced and used by a project team and provides task-oriented access to documents in the repository to reduce the burden of developers, as they need not determine which documents are relevant for their current task. Besides merely making documents available for pull access, the system may actively push important information to its users as soon as it becomes accessible. For example, a manager may be notified when a task gets delayed or an implementer may be notified when an update of a design document becomes available.

Synchronous communication: Synchronous communication like audio and video calls or text chat is available in MILOS via Microsoft NetMeeting. This enables two developers to discuss problems or to perform pair programming (using NetMeeting's application sharing capability).

3.2. Creating a Feedback Loop for Knowledge Maintenance

We believe that maintaining an experience base needs to be tightly integrated with process execution. To support an organizational learning process, our approach links process-centered knowledge management with process enactment support and experience packaging. Figure 1 shows an overview on our approach.

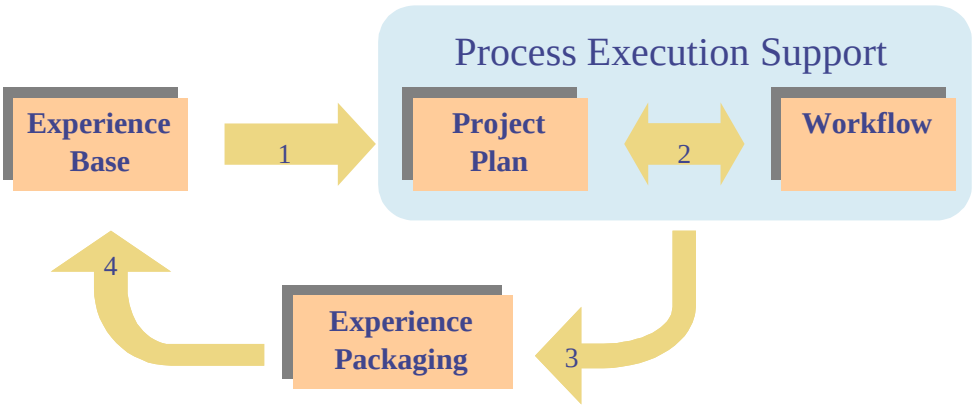


Figure 1. Knowledge Utilization Cycle.

The experience base contains descriptions of best practices in software development in the form of process models. The MILOS experience base is a process model library of generic, reusable process descriptions (called process types): each process type represents a certain class of tasks that frequently occur in the organization's software projects. The library allows capturing knowledge about software development by describing tasks,

potential task decompositions, information flow, and background knowledge. Background knowledge is attached to process types – resulting in a process-centered navigation structure for accessing knowledge.[8] In step ①, a project manager may select processes and methods – decompositions of processes into finer grained processes - from the experience base to create an initial project plan. This automatically links concrete tasks in the project plan to process types from the experience base. For example, a task "inspect the accounting subsystem" from the project plan may be linked to the process type "software inspection" in the experience base. Hence, the plan may include references to background knowledge from the experience base (in this example, e.g. a reference to an "inspection checklist"). The current project plan is the basis for our workflow engine ②. However, the planning process is incremental: we can change the plan at any time during process execution. The next step is to extract reusable process knowledge from the current project plan ③. A user can select parts of the project plan (e.g. a new task that the planner created from-scratch, i.e. a task that was not based on a process type in step ①) and upload/integrate them into the generic process model library ④.

4. Knowledge Management in MILOS

Creating effective knowledge management structures is one of the key factors in software process improvement initiatives (like the Capability Maturity Model, Spice, Trillium, etc.). Specific organizational structures (e.g. a software process group) are developed for the purpose of managing and distributing knowledge about software development.[9] These structures are costly to maintain, and improving their efficiency by a dedicated tool infrastructure seems to be highly desirable. In addition, allowing team members to update the experience base, specifically the contents regarding tutorials on new technologies and background information, can reduce the burden on the process group. The capability of maintaining a shared experience base within a community of practice is especially useful for loosely coupled teams as they will not establish a central process group.

In general, two mainstreams concerning tool infrastructure can be distinguished: first, process-centered software engineering environments (PSEE) [Garg, Jazayari 1996] are acknowledged tools to help in planning, managing and executing today's software projects. Their support is mainly focused on the coordination of the different activities within a project following a defined development process, i.e. focused on project coordination. That is

[8] Our process-centered approach for organizing background knowledge allowed to access *static* background knowledge (e.g. checklists, manuals etc.). It was originally described in [Maurer, Holz 1999]. In the remainder of this paper, we extend it by using run-time information to provide *dynamic* access to knowledge sources.

[9] As in MILOS, the knowledge about the software process is often documented in form of a process model. This model is then described in a set of text documents (sometimes web based) or, more formally, in a process modeling language.

why the support for the individual team member in performing tasks is mainly restricted to provide access to input products for a task, to tools to create output products and to static background information (e.g. checklists etc.).

Secondly, tools that aim at supporting the Experience Factory proposed by [Basili et al 1994] are being developed (see e.g. [Althoff et al 1998]). They encompass mechanisms that make information and knowledge accessible and (re)usable. The main concept here is to document and store any kind of information (e.g., knowledge, experience) and to access it in a structured way. Information retrieval based on statistic document analysis techniques mainly used in document management systems (DMS) is extended by, e.g., similarity-based or ontology-based retrieval, to find more relevant information items and to extract new information facts by the combination of two or more existing items.

The MILOS project aims at providing an infrastructure that integrates these two mainstreams. Its primary goals concerning knowledge management are

- to use information from the task context that is only available when the task is executed, in order to provide more selective access to background knowledge
- to provide dynamic access to relevant task-related knowledge when a task is planned or executed
- to create a feedback loop from process execution to knowledge management resulting in support for learning software teams
- to improve knowledge exchange between members of a virtual team by providing simple search facilities for finding experts in a technology
- to model knowledge needs for tasks to dynamically retrieve up to date information from various knowledge sources residing within a community of practice

Reaching these goals will improve the efficiency of process execution as well reduce the knowledge management effort of the organization. In the following, we will show how these effects are achieved.

4.1. Task-Specific Organization of Knowledge

Current process-centered software engineering environments support planning, coordination and enactment on the project level. So far there has been little or no support for the individual project member when confronted with a specific task. Rather, it is assumed that she already is equipped with all the knowledge relevant to perform the task or that she at least knows how to find that information. However, studies show that people often are not aware of information that might be relevant to them, even though this knowledge has been

explicitly stored in the company's organizational memory [Mahe, Rieu 1997]. Hence, when executing a task, a team member should be pointed to knowledge that is relevant for this task. This is the underlying reason why MILOS structures knowledge according to processes.

Process Models represent knowledge about a software development processes and allow for describing arbitrary development processes. Primarily, these models describe different tasks[10], different ways how to solve a specific task (called *methods*), and the information flow between tasks. For a more detailed description see [Verlage et al 1996].

Process models can be a basis for enactment but they do not support an individual team member in doing their job nor do they support a selective access to pieces of knowledge in a vast repository. Hence, MILOS realizes a process-centered knowledge management approach that attaches additional information to process models:

- ***Skill models***: MILOS allows attaching a skill model to tasks. Skills can be technical skills (e.g. Java programmer), tool knowledge (e.g. knows VisualAge for Java) or general skills (e.g. understands German). Skills are organized in an ontology (a hierarchy of concepts and their relationships). Attaching required skills to a task helps in two ways. First, it supports finding a team member with the right skill set. Second, it allows searching for experts in a specific area and then initiating communication with them[11]. This approach reduces the amount of knowledge that needs to be stored in the experience base because one can "simply ask the expert" if a problems arises. Third, the skill information attached to task types and to a team member can be used to select the right background material to avoid information overload. For example, MILOS can provide a short checklist to an expert tester and a detailed test specification to a novice tester.
- ***Links to background knowledge***: Any kind of Web accessible document[12] can be attached to a process. This may include on-line tutorials, specifications from standards bodies, company manuals, checklists, hard-coded database access calls etc. Team members are easily able to attach additional links to a process while they work on it. Basically, this is like creating a task type specific bookmark or favorite list that is shared between a community of practice (i.e. all developers that are performing tasks of the same type are considered to be 'sharing a practice').

[10] In the context of experience management, task, task type, process and process type are used synonymously.

[11] MILOS includes a resource pool component that manages the skill set of each team member and uses ontology-based retrieval for finding people with a given skill set.

[12] More specifically, MILOS is able to attach any HTTP GET request to a process type as the implementation simply stores an URL.

- ***Information needs models***: While links pointing to background knowledge are useful for providing access to task type specific information, they are not taking concrete project circumstances into account. Hence, we extended MILOS with the ability to model information needs that utilize information that is only available when the task is being executed. For example, when MILOS knows during enactment that the system to be developed is a real-time system developed in Java, it may want to provide access to information about the Java 2 Micro Edition (J2ME) from the JavaSoft web site. Conceptually, this requires to define queries that can contain variables and then bind these variables during enactment to values maintained by the workflow engine (for example, from attribute values attached to processes). The next section elaborates these information needs models in detail.

In summary, a process-centered knowledge management approach has the following advantages:

- Processes/tasks are "natural" entities for managers and team members: they are well used to thinking in processes (e.g. for project planning).
- For their daily work, people don't need knowledge per-se but knowledge that helps them to perform their specific tasks. A process-centered knowledge management system explicitly associates the task with the knowledge needed for it.
- By linking relevant information sources and dynamic queries to a task, the "lost in hyperspace" problem is reduced because the user is actively guided to available knowledge needed in his situation instead of being forced to somehow find relevant information on his own.

4.2. Information Needs Modeling and the Information Assistant

MILOS contains an "Information Assistant" (IA) that realizes an task-sensitive information delivery concept. The concept is based on an explicit model of information needs that might arise during specific tasks, together with an automatic query to the appropriate information source from which the answers can be retrieved

Figure 2 gives an overview of the IA. Process models (PMs) from the experience base are extended by information need objects (INOs). These enriched PMs can be used as templates for a project plan ①, which is interpreted ② by the workflow engine. The Information Assistant supports users carrying out planning or execution activities. It provides information ③ based on available INOs and other background information (see section above). The IA queries information sources on demand to retrieve information ④ and delivers the results to the user ③. From within the IA, a user can post questions that are added to the appropriate process model element ⑤ as part of the feedback loop (see Section 5).

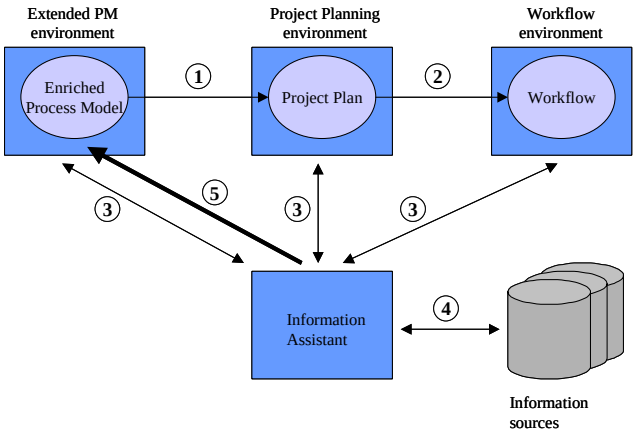


Figure 2. The knowledge delivery concept of MILOS.

INOs represent potential information needs that might arise during planning or enactment of specific processes. Figure 3 shows an example process model where INOs have been attached to different model elements, e.g. to processes, process decompositions (complex methods), process specializations (atomic methods) and parameters. The "*" denotes that several INOs can be associated with a model element.

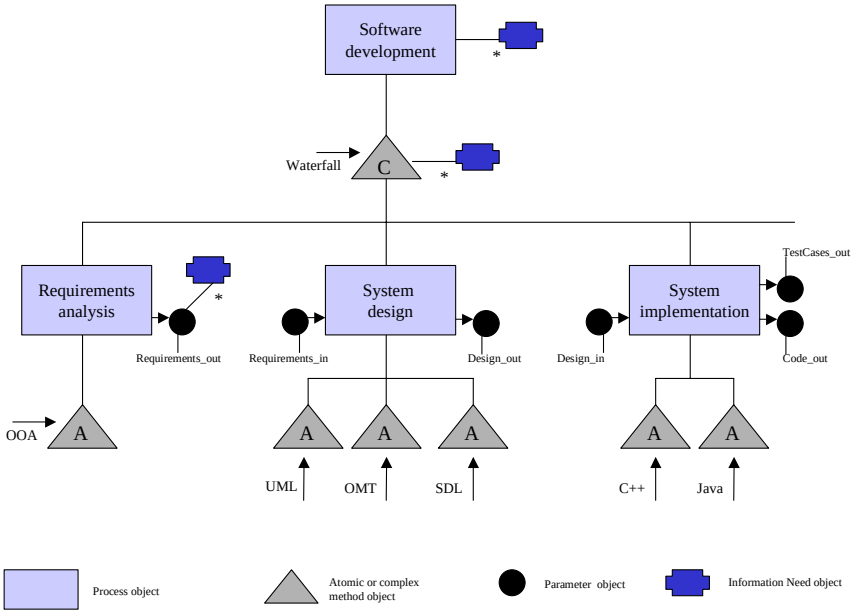


Figure 3. Example process model with information need objects.

INOs are characterized by the following attributes:

- ***Representation***: a textual representation that describes the information need, e.g. "What bugs have been reported for component <componentName>?". The representation can contain variables

(delimited by '<' and '>') that correspond to attributes defined in the process model. During planning or enactment, they will be substituted by their current values.

- ***Information source***: specifies which resource can be used to retrieve information that might be helpful to satisfy the defined information need. Sources can be e.g. bug tracking systems, information agents, databases, or document management systems. A number of typical interfaces (e.g. SQL, CGI, HTTP) have been predefined within MILOS to allow access to different types of information sources.
- ***Query representation***: specifies a query in the syntax required by the given information source, `e.g.: "?bug_status=NEW&product=<component Name>".` The query will be executed automatically to retrieve helpful information items from the information source. Before executing the query, variables are bound to values retrieved from the current task context. This allows for taking into account e.g. the skill level to the team member executing the task or technologies required for this task.
- ***Supported roles***: specifies one or more roles to which the retrieved information will be relevant. For example, that way we can distinguish between information needs typical for project planners or software developers.
- ***Information category***: specifies one or more categories under which the INO will be organized; examples of predefined categories for the role `planner` are: "`agent assignment`" and "`project scheduling`".
- ***Precondition***: a Boolean expression that specifies when the information need normally occurs and the query can be launched, e.g.: "`componentName.hasValue()`"

Thus, a process model not only serves as a means to store knowledge about best *process* practices, but also maintains *dynamic queries* that have been found useful in the past. Both planners and developers are confronted with a set of standard problems that occur for every task. In order to enable the Information Assistant to satisfy these information needs, we require the availability of appropriate information sources from which the desired information can be obtained, e.g.:

(i) the organization's resource pool, including skill information for each agent,

(ii) a calendar database providing schedule dates for agents,

(iii) bug and issue tracking systems,

(iv) project and task case-bases,

(v) an experience base storing process and quality models,

(vi) a document management system holding general documents

(vii) a web search engine

As mentioned above, information needs are represented as INOs that reference an information source. Table 1 shows an example INO that references a task case-base. In order to launch the query execution, the IA must be able to access the task case-base via a predefined interface. In accordance to this interface, it must also replace the variable <currentTask> with the task for which the INO's query execution is requested. In the example of our task case-base, this means replacing the variable with the current task's name string representation. This representation becomes part of a parameter string for a CGI-script that performs the similarity-based search.

INO attributes	**Value**
Representation	"Who has performed similar tasks before?"
Query	retrieve_similar_tasks(<currentTask>)
Information source	Task case-base
Precondition	Task case-base available
Information category	agent assignment, task execution
Supported roles	planner, enactor

Table 1. Example of a generic INO that is associated with every task.

In addition to information needs that typically occur for all tasks, we capture information needs that only occur for certain tasks. Processes in the process model are enriched with INOs as depicted in Figure 4, where we define an INO that reflects a programmer's information need to view the list of bugs that have been reported for the component he is currently implementing. From the tree in the upper-left part of the window, the user has selected the process object 'Implementation of component' from a process model. The tree in the upper-right part displays the associated INOs, using the role and information category attributes for structuring. The attribute values of the selected INO are shown in the lower part of the window.

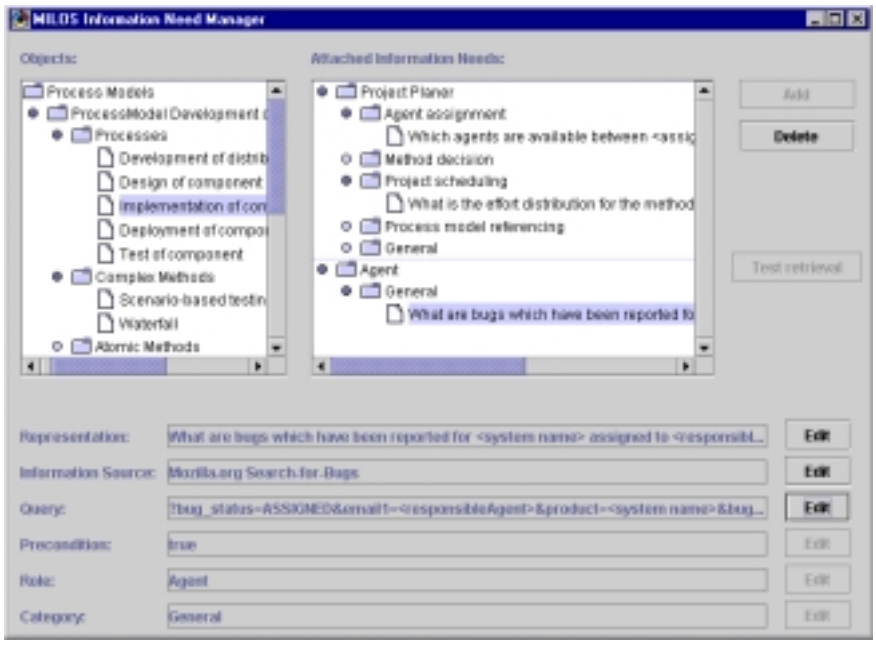


Figure 4. The MILOS INO editor.

We now discuss information needs for project planning as well as for task execution in more detail.

4.3. Using INOs to Support Project Planning

A manager[13] plans by tailoring generic process models to the needs of a concrete project. This includes:

- Selecting processes and methods from the generic process model that should become part of the project plan. For example, the generic process model contains a process "System Design" with methods "RUP", “Agile Modeling” and "SDL". The project manager can add an instance of the process "System Design" to the project plan and then select "RUP" from the set of alternatives as his method of choice.
- Standard project planning: Assigning tasks to responsible agents, scheduling tasks, cost estimation etc. COTS tools like MS Project commonly support these activities.

In order to support planners in these knowledge intensive activities, MILOS provides them with situation-specific information that helps them in their decision-making. Table 2 lists examples of typical information needs in project planning.

[13] If MILOS is used for lightweight processes like Extreme Programming, the whole team instead of a project manager does the planning. Nevertheless, the information needs stay the same.

Category	Information need
Anticipating difficulties	• What problems occurred in similar tasks and how were they solved? • What relevant problems /bugs with the tools used in the current task are known?
Agent assignment	• Which agents match the skills required for the task? • Which agents have performed a similar task before? • Which agents are available at the time period in question?
Effort estimation	• What quality models exist for the task? • What was the effort of similar tasks?
Task refinement	• What standard refinements exist for the task? • How were similar tasks performed in the past?

Table 2. Typical information needs of planners.

When project plans are tailored process models, plan elements inherit the generic INOs associated with the corresponding process model elements. In the context of the current project, variables referenced in the INO definitions can now be substituted by their values (e.g. the variable <assignedAgent> is substituted by the team member to which the task under consideration is currently assigned to). As a consequence, INOs that are intended to reflect a planner's information needs can now be presented to him by the Information Assistant, in order to signal available access to potentially useful information.

4.4. Using INOs in Project Execution

The MILOS workflow engine interprets project plans in order to actively guide human users in their work. In addition, the INOs associated with tasks are presented to those team members on whose to-do lists the entities appear; thus, they are given situation-specific access to potentially relevant information. Table 3 lists typically information needs during task execution.

Category	Information need
Anticipating difficulties	• What problems occurred in similar tasks and how were they solved? • What relevant problems/bugs with the tools used in the current task are known?
Access to tool	• Who is an expert for the tool I use?

related information	• Do newsgroups or mailing lists exist for the tool? • Where is the tool web site? • How does the tool integrate with another? • How do I install another tool component? • Where do I find tutorials?
Access to technology	• Where do I find the specification of this technology? • Who is an expert on this technology? • In which other projects did we use the technology? • Where do I find newsgroups or mailing lists on this technology? • Where do I find tutorials or examples?

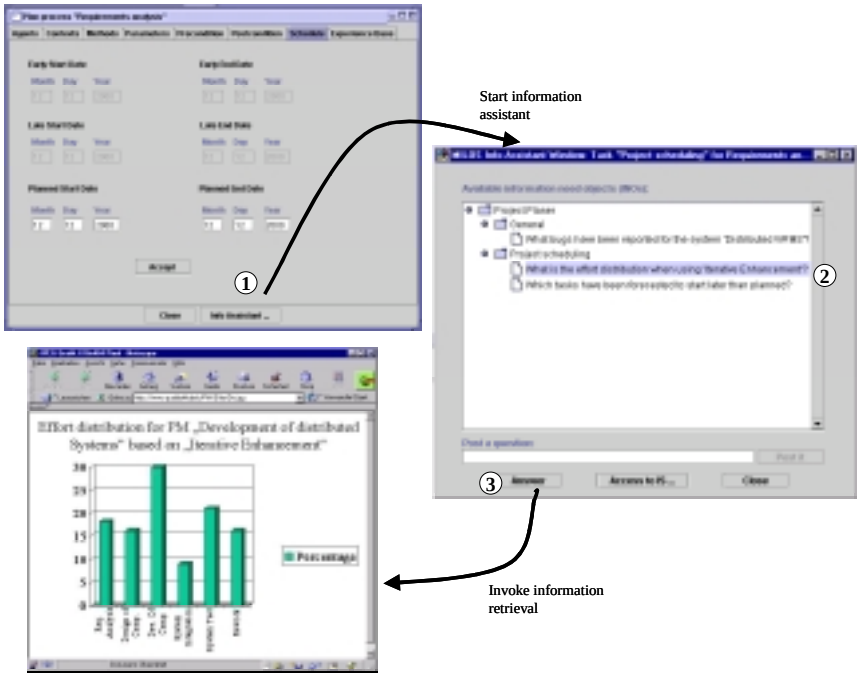
Table 3. Typical information during task execution.

4.5. Information Assistant Scenario

In our example, the project planer starts MILOS to set up a new project, i.e. to define a project plan. For his current project, he provides a characterization by specifying the project name (’Distributed workflow management system’), the system architecture type (’Distributed’), and the estimated duration (’2 years’). Before creating a new project plan, he tries to find similar projects conducted in the past that might help him in his planning. The retrieval results show MILOS project plans that have been stored in a project plan database using case-based reasoning technology (see e.g. [Althoff et al 98]). Inspecting the most similar project plan, he can see that this plan used the process model ’Development of distributed systems’ from the MILOS experience base. Hence, he browses the process model library and selects this process model as the basis for his project plan. As a result, the processes from the experience base now define tasks within the project plan. Besides the specification of these processes, their variables, and the possible methods, the corresponding instances in the project plan include the INOs that are attached to the process model objects.

The next step is defining a schedule. Since the Information Assistant signals possible support, he starts the Assistant from the project planer user interface. The IA presents him the current relevant and available INOs and he selects the question "What is the effort distribution for the process model ’Development of distributed systems’?". As a result, the IA presents him an effort distribution diagram.

Next, the planner wants to estimate the effort required for the task ’Requirements analysis’. He consults a quality model that describes the effort distribution with respect to process steps in the chosen process model. Using the Information Assistant (see Figure 5, ①), he accesses the information associated with the question "What is the effort distribution when using Iterative Enhancement?" which is listed in the information category ’Project scheduling’ ②. A corresponding diagram retrieved from the experience base for quality models is displayed in a window after pressing the “Answer” button ③. According to the effort distribution and the



estimated total time of 2 years for his project, he can now provide rough estimates for all high-level tasks.

Figure 5. The information assistant in project planning.

Furthermore, the planer has to assign responsible team members to each of the tasks specified in the project plan. He wants to do this for a design task for which he has not yet chosen a method. Existing alternatives are ’RUP’ and ’SDL’. He accesses the task assignment user interface. As the project planer is fairly new in his job and the department, he requires information whether any team members are working in the department that have experience in one of these methods. The Information Assistant provides an INO within the information category ’agent assignment’ which models this information need represented as "Which agents have experience with ’RUP’ or ’SDL’?". The question is coupled with the retrieval on a skill database of team members, which yields a list of team members with the required skills. He can see that expertise for both methods exists in the team. He decides to use ’RUP’ as he personally has used this method before and is convinced of its quality. Due to this

decision, his current work context changes as the method decision for the design task gets the value ’RUP’. This is propagated to the Information Assistant that updates his list of INOs: The general question about experienced team members for both methods is not longer required as a decision has been made.

Since he has already provided start and end dates for the design task, he is now interested in team members which have experience with ’RUP’ and are available in the given time frame. Thus he uses the appropriate question from the Information Assistant to query the schedule database for available team members, checking only those team members that have RUP skills.

The project planer knows from formerly planned projects that another method for design is ’OMT’. Besides the planning for the current project he is interested if any team members in the new department have experience with this method. As ’OMT’ is not defined in the experience base, the Information Assistant does not offer support for this question. But as the retrieval for ’RUP’ and ’SDL’ has been done on a skill database, he uses the Assistant to access the query engine interface of this information source. Now he defines his own query to search for agents with ’OMT’ experience and launches it. As he thinks this information need might occur again in the future (maybe even the ’OMT’ method can be modelled as additional method in the process model) he starts the INO Manager within MILOS and adds a corresponding INO to the INO list of the ’Design’ process into the information category ’Agent assignment’. From now on, this INO appears whenever someone uses the IA while doing agent assignment for the ’Design’ task.

Likewise, as with the agent assignment for the ’Design’ task, the project planner proceeds with the remaining tasks in the project plan.

4.6. INO execution to find an EJB tutorial

Team members that participate in the project can access individual to-do lists. The project planer has assigned a task ’Implement WFE component’ during project planning to ’Alice’ and, accordingly, this task appears in her to-do list (see Figure 6).

When she starts working on the task ’Implement WFE component’, she runs into a problem while trying to implement an Enterprise Java Bean (EJB). She remembers that this once has been explained to her in a tutorial. She launches the Information Assistant again and identifies her question "Where can I find a tutorial on EJB?” in the presented list, which is exactly what she is looking for. She selects the INO and executes the query (as shown in Figure 6) to retrieve a list of items found on the JavaSoft web site. The JavaSoft web site has been

known as a good knowledge source by the one who modelled the INO. She follows the hyperlink to the EJB tutorial, in order to refresh her knowledge while following the tutorial steps.

A further look in the Information Assistant window guides her to the question "Which experience has been documented for Java/EJB implementation?". She does not know that any experience is documented related to that, but seemingly these documents exist as otherwise this information need would not appear in the Information Assistant. Since there are documented experiences, there must also be people who have been through these experiences. As she cannot find a corresponding information need in the presented list of questions (i.e. INOs), she uses the posting functionality of the Information Assistant to state her question and post it. The Information Assistant attaches the question to the process object in the project plan that defines the task 'Implement WFE component' and notifies the project planer (i.e. manager) that an information need has occurred that has not been satisfied with the given INO specifications. The manager can now initiate further actions, either to provide the required information for agent 'Alice' and/or to model this information need as a persistent INO.

In summary, besides having used the provided information need support via the IA, the project planer has modelled at least one new INO about the 'OMT' method and Alice has posted a question which has been attached to the project plan. When a project is finished, the process model can be written back into the MILOS experience base. Using this technique, the newly modelled INOs and the posted questions become persistent, here as part of the process model 'Development of distributed systems'. They are available in the future every time this process model is reused. The posted questions need to be modelled into a completely specified INO (e.g. linked to an information source) to be usable in the future. Before that, they serve as a marker for an INO modeller that an information need occurred for the specific process model object to which it is attached.

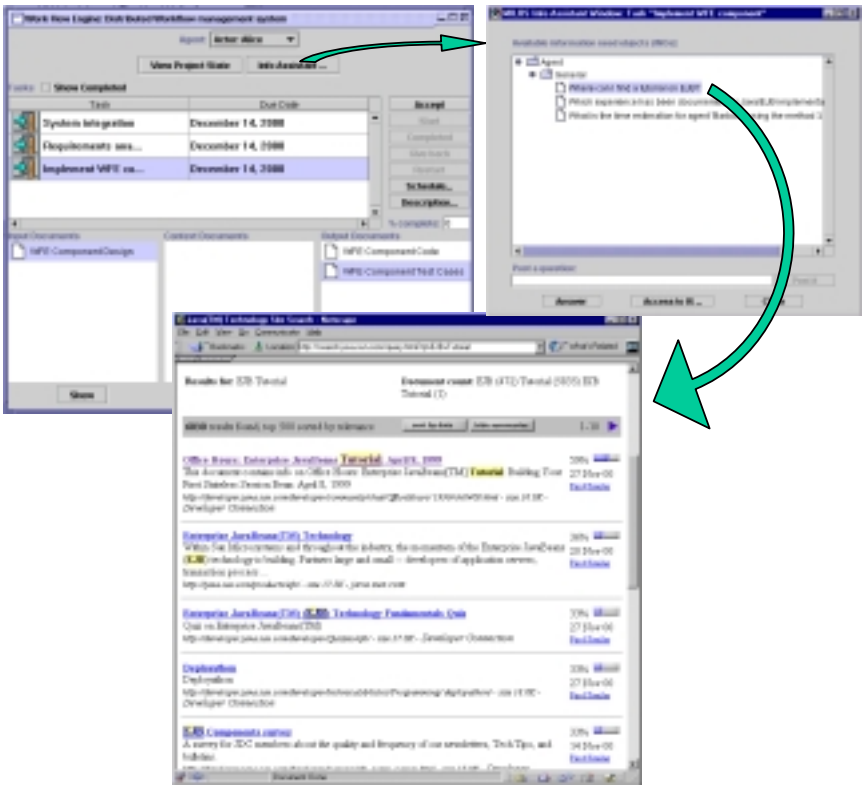


Figure 6. Using the Information Assistant during task execution.

5. Supporting Organizational Learning

MILOS supports the feedback loop that is required for creating a learning software organization in several ways:

- ***Uploading project plans to the experience base***: Team members can upload parts of the current project plan to the experience base. This process basically copies a process, its selected subprocesses and the corresponding information flow and background links to the experience base. There it can be edited to make the names of the tasks generic.
- ***Adding new INOs***: As shown in the example, team members can add new INOs (or, at least, requests to produce them) to processes in the experience base.
- ***Task-type specific discussion groups***: The experience base may provide access to task-type specific newsgroups (e.g. a newsgroup dealing with black-box testing). Team members can post questions dealing with the execution or planning of this task type to the newsgroup. Other team members or members of the organization's process group may answer these questions. As news messages are

archived, these dialogues become part of the experience base and may be used in the future by other team members for finding answers to their questions.

- ***Skill/technologies/tool specific newsgroups***: The experience base may also provide access to newsgroups discussing specific skills/technologies/tools. They work in a similar way as task-type specific newsgroups.
- ***Creating FAQs***: The process group can analyze the discussions in the newsgroups from time to time to determine reoccurring questions. To avoid unnecessary overhead in reading irrelevant messages in a newsgroup, these reoccurring questions can then be put into a FAQ document that is then linked to the corresponding task type in the experience base.
- ***Evaluation of usefulness of information***: An extension of MILOS will allow team members to evaluate the usefulness of the attached information or INOs. This feedback can then be used for updating the experience base and also for sorting the lists that allow users to access background knowledge.

Using the feedback loop, process models that are frequently used are more or less automatically maintained. They provide better support each time they are tailored for a project. The MILOS approach thus offers tool support for creating a learning software organisation as it helps to package required information and provides it at the right time (i.e. while planning or enacting a particular task) to the right team member.

5.1. Supporting Organizational Learning in Virtual Teams

While any software development team can use the MILOS approach, it is specifically useful for virtual teams. In virtual teams, members frequently change. Hence, there is a high demand on bringing new members up to speed on their tasks and in preserving good sources of knowledge for the organization.

Knowledge sharing between co-located team members is most easily done by informally talking to each other, for example in a coffee break or over lunch. In such a situation, there is less need for tool support. Virtual teams, on the other hand, do not have this option. Their members are not co-located and are often distributed over different time zones. This limits the amount of synchronous communication and increases the need for tool support.

The information needs modelling approach realized by MILOS allows creating cross-project communities of practice by connecting people working in different projects on the same task type (e.g. “RUP”) or using the same technologies (e.g. “EJB”). As software development often has to struggle with fast changing technology,

keeping the contents of an experience base up to date is a demanding task and needs to be integrated as much as possible with the everyday processes of executing processes.

6. State of Implementation

Two prototypical implementations of the MILOS approach exist. One is based on the object-oriented database management system GemStone, the other uses relational databases and Enterprise Java Beans. Both implement the process support environment. The GemStone version includes a prototypical implementation of the Information Need Modeling and Information Assistant components. The EJB version is Web-based and accessible at the MILOS web site[14].

7. Related Work

Related work comes mainly from two areas: Software Process Improvement and Knowledge Management. We first discuss software process improvement and then we analyze some KM approaches.

Most process improvement approaches, e.g. capability maturity model, SPICE, QIP, require describing the development processes more or less formally. Within the framework of software process modeling, several languages were developed that allow for describing software development activities formally [Osterweil 1987; Curtis et al 1992; Armitage & Kellner 1994]

Software process models represent knowledge about software development. They describe activities to be carried out in software development as well as the products to be created and the resources & tools used. These models can be a basis for continuous organizational learning as well as the actual basis for the coordination and the management of software engineering activities.

Software process modeling and enactment is one of the main areas in software engineering research. Several frameworks have been developed (e.g. procedural [Osterweil 1987], rule-based [Kaiser & Feiler 1988; Peuschel et al 1992], Petri net based [Bandinelli et al 1993], object-oriented [Conradi et al 1994].

Managing software process knowledge is also the goal of the experience factory approach [Basili et al 1994]. They distinguish between the organizational structure to manage the software knowledge (the experience factory department) and the activities that have to be carried out to build an experience factory.

[14] http://sern.ucalgary.ca/~milos

MILOS extends existing process modeling approaches by information need models that use execution time information to bind variables in dynamic queries. As a result, retrieval results are situation specific and reduce the information overload of team members.

Knowledge management and organizational memory is currently a hot topic in research (for example, see [Davenport et al 1997; Wielinga et al 1997]). The use of case-based reasoning technology for experience management in software engineering is discussed in [Althoff et al 1998]. Their approach is not directly linked to project execution support and therefore fails in providing an operational feedback loop between project execution and organizational learning. Their approach could be easily integrated with MILOS by pointing our CBR queries to their system.

The expert system group at the German DFKI is also following a process-centered approach to knowledge management. Their tool is based on a business process modeling approach and is, compared to MILOS, fairly inflexible at enactment time: changing the process by simply replanning the project with a COTS tool is not supported. On the other hand, their approach is based on domain ontologies [Kühn & Abecker 1997] that allow for a semantic-oriented information retrieval.

Ontology-based retrieval is also investigated by [Decker et al 1999]. Ontology-based KM approaches are not providing a process-centered approach and therefore require the user to specify queries for a given task (whereas MILOS can provide users with predefined queries for tasks). A similar argument holds for hypertext-based approaches (e.g. [Euzenat 1996]): they do not provide task-oriented access to knowledge but force the user to navigate to it.

8. Conclusion and Future Work

In this paper, we described our approach for supporting virtual software development teams in organizational learning. While virtual teams have their drawbacks [Herbsleb, Grinter 1999; Herbsleb et al. 2001], they are simply becoming more and more commonplace. Hence, the question is not if virtual teams are better than co-located teams but the question is how can we support virtual teams more effectively.

Our approach integrates process execution with knowledge management to create an organizational feedback loop. The work presented here extends existing approaches in two ways:

- Information need models are a lightweight approach for querying heterogeneous information sources while using situation-specific data (that only exists in the context of the current project execution). While existing process modeling approaches describe a software process statically, information needs

models allow to formulate dynamic queries for team members involved in project execution. Executing information needs models when the project is carried out provides knowledge to team members for doing their job.

- Concrete project plans and new information needs that arise through project execution can be fed back into the experience base and then later reused in other projects. This direct-feedback approach supports organizational learning in virtual teams and the creation of communities of practice.

Whereas undoubtedly the introduction of new tools at first results in an increased workload, we argue that, in the long run, the proposed approach will reduce work. In particular, the Information Assistant will relieve the process group from answering standard questions in the same way as help-desk applications already have proven to do in other domains (e.g. trouble-shooting for technical devices). We expect the Information Assistant to provide answers to the users' standard questions (especially to those of new employees), so that the human experts need only be consulted for new, more difficult problems.

The tight integration of process execution support and knowledge management makes a good base for software guilds. Reoccurring information needs can trigger educational programs. The on-line collaboration facilities and on-line experience base may provide access to useful knowledge for guild members and allow developing a virtual community of practice where members help each other in solving problems.

Process modeling approaches usually are seen as heavyweight methods and in direct opposition to agile processes like Extreme Programming [Beck 2000; Beck, Fowler 2000], Scrum [Beedle, Schwaber 2001], Agile Software Development [Cockburn 2002], Adaptive Software Development [Highsmith 2000], Feature-Driven Development [Coad et al. 1999], and DSDM [Stapleton 1997]. In our opinion, our approach is - in fact - conceptually aligned with agile methods[15] as it tries to be "minimally invasive": the goal is to provide "barely enough" tool support for process execution and organizational learning in virtual teams[16]. In MILOS, process Models are not restrictive; rather, they represent templates that can be enriched with information need models. While virtual teams have to rely more on tool support than, e.g., co-located 'Extreme Programming' teams, this is mainly a result from the fact that these teams are not co-located and need to augment face-to-face communication with some means to support project coordination and knowledge sharing. Our approach to knowledge management focuses on communities of practice and information need models. While communities

[15] In fact, we are currently adapting MILOS to directly support distributed Extreme Programming. A demo of MILOS DXP is available at http://ebe.cpsc.ucalgary.ca/MilosWEB/.

[16] The approach presented in this paper where project planning relies on a process model is only one way how MILOS can be used. Another way is using MILOS as a simple to-do list for a virtual team.

of practice are trying to bring the right people together for problem solving, information needs models try to strike a balance between expressive power and modeling overhead. In the future, we plan to apply and evaluate the MILOS approach in virtual teams using agile software processes as well as more traditional ones.

While the information needs modeling approach is reducing the information overload of developers that search for background knowledge, it still may find more information than required in a given situation. This steams from the fact that all knowledge sources are seen as equivalent. In the future, we are planning to incorporate collaborative filtering techniques to heuristically rank different sources.

9. Acknowledgements

The work on MILOS was supported by the DFG (as part of SFB 501: “Development of large systems with generic methods”), DLR, NSERC, ASRA, The University Of Calgary, and Nortel with several research grants.

References

Althoff, K.-D. & Bomarius, F. and Tautz, C. (1998), "Using Case-Based Reasoning Technology to Build Learning Software Organizations", In *Proceedings of the 1st Interdisciplinary Workshop on Building, Maintaining, and Using Organizational Memories (OM-98)*, 13th European Conference on AI (ECAI’98), Brighton, http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-14/ .

Apache Foundation (1999), "About the Apache HTTP Server Project", http://httpd.apache.org/ABOUT_APACHE.html.

Apache Foundation (2001), "The Jakarta Site –Reference Library", http://jakarta.apache.org/site/library.html.

Armitage, J. and Kellner, M. (1994), "A conceptual schema for process definitions and models", In *Proceedings of the Third International Conference on the Software Process*, IEEE Computer Society Press, Los Alamitos, CA.

Bandinelli, S., Fuggetta, A. and Grigolli, S. (1993), "Process Modeling-in-the-large with SLANG", In *Proceedings of the Second International Conference on the Software Process*, IEEE Computer Society Press, Los Alamitos, CA.

Basili, V. R., Caldiera, G. and Rombach, H. D. (1994), "Experience Factory", In *Encyclopedia of Software Engineering, vol. 1*, J. J. Marciniak, Ed., John Wiley Sons.

Beck, K (2000), *Extreme Programming Explained - Embrace Change*, Addison Wesley, Reading, MA.

Beck, K., Fowler, F. (2000), *Planning Extreme Programming*, Addison Wesley, Reading, MA.

Beedle, M., Schwaber, K. (2001), *Agile Software Development with SCRUM*, Prentice Hall, Englewood Cliffs, NJ.

Coad, P. et al. (1999), *Java Modeling Color with UML*, Prentice Hall, Englewood Cliffs, NJ.

Cockburn, A. (2002), *Agile Software Development*, Addison Wesley, Reading, MA.

Conradi, R., Hagaseth, M., Larsen, J. O., Nguyen, M., Munch, G., Westby, P. and Zhu, W. (1994), "EPOS: Object-Oriented and Cooperative Process Modeling", In *PROMOTER book: Software Process Modeling and Technology*, Finkelstein, A., Kramer, J. and Nuseibeh, B.A. (Eds.), Advanced Software Development Series, Research Studies Press Ltd. (John Wiley).

Curtis, B. and Kellner, M., Over, J. (1992), "Process modeling", *Communications of the ACM*, 35(9), 75–90.

CVS (2001), "Open Source Version Control Software", http://cvshome.org/.

Davenport, T.H., Jarvenpaa, S.L. and Beers, M.C. (1997), "Improving Knowledge Work Processes", *Sloan Management Review*, 37 (4):53-65, 1997.

Decker, S., Erdmann, M., Fensel, D. and Studer, R.(1999), "Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information", In *Semantic Issues in Multimedia Systems*, R. Meersman et al. (eds.), Kluwer Academic Publisher, Boston.

Dellen, B., Kohler, K. and Maurer, F. (1996), "Integrating Software Process Models and Design Rationales", In *Proceedings of Knowledge-Based Software Engineering Conference (KBSE-96)*, IEEE press.

Euzenat, J. (1996), "Corporate Memory through Cooperative Creation of Knowledge Bases and Hyper-documents", In *Proceedings of the 10th Knowledge Acquisition, Modeling and Management for Knowledge-based Systems Workshop (KAW'96)*, Banff, Canada.

Garg, P.K. and Jazayeri, M. (1996), "Process-centered Software Engineering Environments", IEEE Computer Society Press, 1996.

Herbsleb, J. D., Grinter, R. E. (1999), "Splitting the organization and integrating the Code: Conway;s law revisited", In *Proceedings of the 21th International Conference on Software Engineering (ICSE 1999)*, IEEE Computer Society Press, Los Alamitos, CA.

Herbsleb, J. D., Mockus, A., Finholt, T. A., Grinter, R. (2001), "An empirical study of global software development: distance and speed", In *Proceedings of the 23th International Conference on Software Engineering (ICSE 2001)*, IEEE Computer Society Press, Los Alamitos, CA.

Highsmith III, J.A. (2000), *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, Dorset House Publishing.

Kaiser, G. E., Feiler, P. H. and Popovich, S. S. (1988), "Intelligent Assistance for Software Development and Maintenance", IEEE Software.

Kühn, O. and Abecker, A. (1997), "Corporate Memories for Knowledge Management in Industrial Practice: Prospects and Challenges", In *Journal of Universal Computer Science* 3, 8, Springer Science Online. URL: http://www.iicm.edu/jucs_3_8/ corporate_memories_for_knowledge.

Laubacher, Robert J. and Malone, Thomas W. (1997), "Flexible Work Arrangements and 21st Century Worker's Guilds, Initiative on Inventing the Organizations of the 21st Century", Working Paper #004, Sloan School of Management, Massachusetts Institute of Technology, October 1997, http://ccs.mit.edu/21c/21CWP004.html

Malone, Thomas W. and Laubacher, Robert J. (1998), "The Dawn of the E-Lance Economy", *Harvard Business Review*, Sep-Oct 1998.

Maurer, F., Dellen, B, Bendeck, F., Goldmann, S., Holz, H., Kötting, B. and Schaaf, M. (2000), "Merging Project Planning and Web-Enabled Dynamic Workflow Technologies", *IEEE Internet Computing*, May/June 2000, pp. 65-74.

Maurer, F., Holz, H. (1999), "Process-oriented Knowledge Management for Learning Software Organizations", In *Proceedings Knowledge Acquisition Workshop 1999*, Banff, Canada.

Mahe, S. and Rieu, C. (1997), "Towards a Pull-Approach of KM for Improving Enterprise Flexibility Responsiveness: A Necessary First Step for Introducing Knowledge Management in Small and Medium Enterprises", In *Proceedings of the International Symposium on Management of Industrial and Corporate Knowledge (ISMICK '97)*, Compiegne, 1997.

Osterweil, L. (1987), "Software Processes are Software Too", In *Proceedings of the Ninth Int. Conf. of Software Engineering*, Monterey CA, pp. 2-13.

Peuschel, P., Schäfer, W. and Wolf, S. (1992), "A Knowledge-based Software Development Environment Supporting Cooperative Work", In *International Journal on Software Engineering and Knowledge Engineering*, 2(1).

Stapleton, J. (1997), *DSDM Dynamic Systems Development Method*, Addison Wesley, Reading, MA.

Verlage, M., Dellen, B., Maurer, F. and Münch, J. (1996), "A synthesis of two software process support approaches", In *Proceedings 8th Software & Engineering and Knowledge Engineering (SEKE-96)*, USA.

Wielinga, B.J., Sandberg, J. and Schreiber, G. (1997), "Methods and Techniques for Knowledge Management: What has Knowledge Engineering to Offer", *Expert Systems with Applications* 13, 1, 73-84.