# An Internet Based Software Process Management Environment[1]

Frank Maurer
University of Calgary
maurer@cpsc.ucalgary.ca

Barbara Dellen
University of Kaiserslautern
dellen@informatik.uni-kl.de

# 1 Abstract

The paper presents a process-oriented view on knowledge management in software development. We describe requirements on knowledge management systems from a process-oriented perspective, introduce a process modeling language MILOS and its use for knowledge management. Then we explain how a process-oriented knowledge management system can be implemented using advanced but available information technologies.

# 2 Introduction

Work is goal directed and process-oriented: Companies have goals to be reached and procedures to follow in pursuing these goals. The more efficient these procedures & processes are, the better the company is able to prosper in the global market.

Improving the efficiency of software processes results in a reduction of time needed to perform the task and of costs. This business objective leads to approaches as *lean management* and *business process reengineering & optimization*. In order to fulfill these objectives, workflow management approaches often are introduced in the enterprise [GH-95]. The workflow management coalition defines workflow management as:

"The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules." [WFMC-96].

[1] This paper is a slightly extended version of the paper "A Concept for an Internet-Based Process-Oriented Knowledge Management Environment", submitted to the KAW 98, Banff, Canada, April 1998.

Today, workflow management approaches are basically only used to support repetitive administrative tasks whereas knowledge-intensive tasks are not supported. There are several reasons for this:

- The knowledge needed for executing the process is not explicitly described in the workflow model.
- Current workflow approaches are not flexible enough to adapt on the fly to changing processes. Our work in the past years was addressing this problem.

Although workflow management has its limitations, it provides a process-oriented view on software development. Compared to most KBS approaches that focus on knowledge representation instead of the process, this is an advantage. KBS approaches on the other side support the acquisition, structuring, and representation of knowledge. They can be used to build up knowledge bases that are then incorporated into the software processes.

## 2.1 Managing the knowledge for software processes

Software development is a knowledge-intensive process where highly educated people have to cooperate to reach the business goal.

Although there were tremendous improvements in software engineering over the last decades, basic problems remain:

- software development is expensive and time consuming
- software projects often run out of time and over budget
- time and cost estimations are error prune

Due to recent developments (e.g. the widespread use of Internet technology), the pressure on software engineers to deliver their products faster increases. These requirements lead to larger development teams that often will be globally distributed. Major problems then are

- how to coordinate and manage the distributed work of the software engineers
- how to provide access to a project's knowledge
- how to manage the experience gathered during past projects and make them usable for the new ones

Workflow approaches for software development are the area of software process modeling and enactment research. Technologies for supporting software development processes are often based on the framework of software process modeling [Ost-87, CK-92, AK-94, Ver-94]. Software process models describe the activities carried out in software development as well as the products to be created and the resources & tools used. The models are a basis for a continuous improvement process (using the Capability Maturity Model) as well as the actual basis for the coordination and the management of the software engineering activities [FH-93].

Understanding commonalties and differences between process types is a key factor for better process support. Process support includes improved communication, guiding people when performing processes, improving both processes and their results, and automating process steps [RV-95].

Software process modeling and enactment is one of the main areas in software engineering research. Several frameworks have been developed (e.g. procedural [SOH-95], rule-based [KFP-88, TKP-94, PSW-92], Petri net based [BFG-93a], object-oriented [CHL-94]).

Process-sensitive software engineering environments which support evolution of executed process models [CFF-93] are a focal point of current research, but the results are still immature [MP-93]. Several approaches to support the evolution and flexibility of software processes were developed [BFG-93b, BK-93, CNG-95, PEM-95]. None of these approaches is supporting globally distributed software processes and their evolution (although work in this direction is starting [Kai-97, Con-97]).

Published approaches on process evolution mainly deal with changing the enacted process model so that it reflects the changed real world process. Automatically sending notifications about the changed process/products over the Internet to the appropriate members of the global team is not supported. To determine them, the system needs a kind of understanding of the causal dependencies between processes and products. Software change impact analysis [BA-96] deals with causal relation between and in products.

Managing software process knowledge is also the goal of the experience factory approach [Bas-89, BCR-94]. 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.

In this paper, we give a process-oriented view on knowledge-intensive tasks in software development. In section 3, we define requirements on process-oriented knowledge management systems. Section 4 discusses our process modeling language MILOS and explains how knowledge is linked to generic software processes. The next section shows how this knowledge can be incorporated into concrete projects. A concept for the architecture of a process-oriented knowledge management system is described in Section 6. Section 7 gives an overview on the state of implementation. In Section 8 we discuss related work. The last section gives a summary of the paper.

# 3 Requirements on Process-Oriented Knowledge Management Systems

In the future, the development of large software systems will be a typical task for virtual enterprises. People with different knowledge and educational background (e.g. economics, computer science, arts for interface design) will work on many locations all over the world (including emerging market countries like India or Eastern Europe) and will have to work together to fulfill the business needs.

Globally distributed work processes have to deal with many problems arising from

- different languages & ontologies
- cultural differences
- different time zones
- different work ethics

- different legal systems
- different hard- and software requirements

Although these problems have sociological causes, some of them can be reduced or overcome using advanced technologies. Some requirements imposed on software processes support systems for the global virtual software enterprise are listed and briefly discussed in the following.

*Requirement 1: Synchronous work support*

Around the clock development requires that - at the end of the working day - the results of the day have to be efficiently communicated to the co-worker in the next time zone who just starts to work. This poses the requirement on modeling approaches, that models have to be easily understood and that an overview on the changes since the last day has to be provided. Nevertheless, synchronous communication between the team members will help in communicating the current state of the work. Audio and video conferencing capabilities can be used as well as shared workspaces and distributed meeting rooms. The technology for that is available but the costs for a wide spread use are still too high.

*Requirement 2: Asynchronous work support*

In globally distributed software development processes, people work asynchronously in different locations and at different times. Therefore, a support system shall facilitate the coordination and the management of the process. Process enactment support techniques - or, in a broader sense, workflow management techniques - try to provide the right information to the right people at the right time with the right tools. They have to be extended to support work over wide area networks.

*Requirement 3: Ubiquitous communication infrastructure*

To reduce the problems in setting up a global team, a ubiquitous infrastructure has to be used. Every team member has to be able to connect to the development process without effort.

*Requirement 4: Transparent & fast access to process knowledge*

For a smooth development process, every team member needs easy access to all relevant project data (e.g. to-do lists, source code, requirement specifications, defect reports etc.) as well as generic knowledge (e.g. coding & documentation standards, effort estimation guidelines etc.). This knowledge can be represented for human use or for use by KBS interpreters. Project data has to be defined in a product model (which imposes a requirement on modeling tools to be open for external access). Using WWW techniques, the access to data is transparent to the user: It does not matter where the data is stored. Clearly, due to current bandwidth limitations, this is not the naked truth: Accessing the data on a foreign server often needs much more (sometimes prohibitive) time than accessing local information.

*Requirement 5: Distributed configuration management*

Configuration management systems are used to maintain several versions of a system and all the information related to it. They are a basis for an orderly development process. Consequently, they have to be extended for the virtual environment of globally distributed teams.

*Requirement 6: Repository for ontologies*

To overcome problems with different ontologies, a project repository has to be provided that defines commonly used terms. Using the Web as the medium, the repository should be organized and maintained at several locations. The repository should be extended to provide an experience base for software development storing generic process and product models as well as metrics gathered in past projects. These could then be used to improve cost and time estimations for new projects.

*Requirement 7: Flexibility of development processes*

Most software process support environments require a complete, fine-grained process model *before* the execution starts. For large-scale software projects, that is not realistic: The project plan needs to be refined and extended while the development is already in progress. In addition, there will be no central project plan in globally distributed projects but the will be plans at every location which have to be coordinated. Using and extending knowledge-based techniques, project planning and execution can be interlinked giving a greater flexibility to the people involved in the project. Agent-oriented approaches may help to coordinate the plans of different locations.

*Requirement 8: Proactive change notifications*

The coordination of globally distributed processes will be improved if part of the task is done automatically. Coordination problems often result from changes introduced into the process because of new external requirements and/or erroneous assumptions. Explicit causal relations between process information generate traceability and can be used by a system to proactively send notifications to team members whose work is influenced by the change (e.g. a task may become obsolete, interfaces of imported modules are changed and users of the interface get notifications). To use change notification mechanisms in worldwide distributed projects; "real" push mechanisms have to be developed. A process enactment engine has to generate events and allow clients to create event listeners.

# 4 The Process Modeling Language MILOS

To fulfill the stated requirements, we developed the new process modeling language MILOS and are in the process of adapting our CoMo-Kit process engine to the new language. MILOS was developed in cooperation with the working group of Prof. Rombach and integrates basic process modeling and knowledge management concepts.

MILOS allows to represent knowledge about software development processes. The core notion of MILOS is the *process*. Any other information is grouped around this notion: Products are inputs or outputs for processes. Factual knowledge is linked to processes. In this sense, MILOS supports a *process-centered structuring* of knowledge.

Knowledge needed to plan and execute includes

- Process, product and resource models
- Project plans & schedules
- Products developed within projects
- Project traces
- Background knowledge such as guidelines, business rules, studies etc.

MILOS offers several language concepts for defining measurement based process models, object oriented product models and resource models. In the following we give a short overview over the main MILOS concepts.

With MILOS product & resource models can be developed and integrated into process models. For a new project, these kinds of models are the basis for the definition of project plans.

## 4.1 Product models

MILOS allows the specification of hierarchical, object-oriented product models. A *product type* defines the structure of a set of products with the same behavior. A product type is either basic or complex. Basic types are predefined and may be integer, real, string, text, date, or external references such as a www page URLs or word documents. A complex type consists of one or more typed subproducts and attributes. Complex product types define hierarchical type structures. Complex product types can be specialized (IS-A relation).

Based on a given product model, products that contain general and specific project knowledge of a company can be specified. The underlying product model defines the structure and the type of that knowledge. A main mechanism for associating knowledge to entities in the process model is using external references to the knowledge. This is further explained in the Section 6.1.

## 4.3 Process Models

Within process models activities and their interrelationship are described.

A process is defined by a description of the process goal, a set of conditions, process attributes, the products needed to plan and to execute the process, a set of alternative methods to reach the process goal, the products to be produced and resource allocations.

Methods are either *complex* or *atomic*. Complex methods refine a process into one ore more subprocesses whereas the application of an atomic method results in the production of products that are the outputs of the process.

Inputs of a process may either be products, that are produced by other processes during project enactment or predefined products taken from generic process models. Here the process view and the knowledge view in MILOS are integrated.

Process models are generic descriptions of the general way of the course of projects. The models have to be specialized and customized within the projects.

Every process is associated with a set of roles and qualifications which are needed to perform the task (e.g. the process “Implement Class” is associated with the qualifications “Java knowledge available” and the role “Programmer”).

## 4.4 Resource Models

Resource models allow assigning roles and qualifications to project team members. Roles and qualifications can be specialized. These models describe knowledge needed by project managers to find appropriate people for a task.

In Table 1 you can find a summary of the basic language concepts of MILOS.

| Concept | Description |
|---|---|

| | |
|---|---|
| process | Set of activities that have to be executed in order to reach a given goal. |
| condition | A condition controls the execution of a project plan. We distinguish between preconditions and postconditions. |
| product type | The description of type and structure of a class of products. |
| product | A product is an information unit of a given product type, for example a document or a piece of code. |
| product reference | Product references stand for the type of products that are used and/or produced by a process or a method. We distinguish between products that are consumed for planning, consumed for execution, produced, and modified. |
| produce | This parameter type stands for a product of a given type that is produced by an atomic method of a process. |
| consume for planning | Product reference that describes the read only access to a product within the planning of a process. |
| consume for execution | Product reference that describes the read only access to a product during the execution of a process. |
| product mapping | Defines the product flow within the process/method hierarchy. |
| method | Problem solving method to reach a process goal. |
| atomic method | Leaf within the process/method hierarchy. It produces or modifies a product. |
| complex method | Problem solving method that refines a process into one or more subprocesses. |
| attribute | Attributes are properties of processes, products, methods and resources. |
| product attribute | Attribute of a product. |
| process attribute | Attribute of a process. |
| resource | Resources are used for the execution of the project. We distinguish between two types of resources: agents and tools. |
| resource attribute | Attribute of a resource. |
| role | A predefined resource attribute. It describes the task of a resource within an organization. |
| qualification | A predefined resource attribute. It describes a skill of a resource. |
| tool | A program that supports activities. |
| agent | A human or machine that carries out activities within a process. |
| precondition | A condition that has to be valid to carry out a process. |
| postcondition | A condition that has to be valid after a process has been executed. |

Table 1: MILOS's representation primitives

# 5 Project Plans: Customized Process Models

Process models are generic, reusable descriptions how to execute projects and are part of the organizational knowledge. They may be for instance stored within an experience factory. For a given project these models have to be adapted to project specific needs. We call the project specific models *project plans*. A project plan can be composed of some different process models. For instance, the test processes of a project plan are taken over from a model for testing, whereas the general course of action is taken over from a waterfall model.

Customizing process models to project plans is part of project planning. Using generic process models, even inexperienced project managers are able to come up with a plan according to the company's quality standards and procedures. In general, the project starts using an initial plan that defines the general course of action and the first project steps. Customizing extends over the project start. On base of the results of early project activities (for example information gathering activities) parts of the plan are further customized during project execution.

We developed techniques to support (a) plan refinement (b) plan adaptation and (c) error correction during project execution. Methods define possible plan refinements. Selecting a method during execution results in a refined plan. Plan adaptation takes place when the definition

of the plan is changed, for example when requirements change or within optimization tasks. Plan adaptation allows, for example, to add new processes to the plan, change the process decomposition, change the order of process execution, and change task delegations.

A least, planning errors may be detected and have to be corrected if the project is already in execution. Changing the plan may affect project execution, if the changed parts of the plan have already been executed and work has been done. For this, our system supports project execution by

- identifying affected processes, results (products) and resources,
- informing affected developers and manager about the change,
- informing them about further consequences of the change,
- tracing the reasons for the change, and
- (automatically) returning to a consistent project state.

Based on AI planning techniques, we developed execution mechanisms that allow to change planning decisions, that propagate the effects of changes through the plan and that notify affected project members. For a detailed description see for [DMP-97].

Changes and adaptations may be located within (local) project plan as well as in the process and product models. This causes two problems:

- When should the change in a process model result in the change of derived project plans? [BK-95] propose in their work a solution of the problem for a reduced set of modeling concepts.
- How to identify the parts of a derived project plan, that are affected from the change, especially if the project plan widely differs from the original process model.

Determining if a concrete project plan (or parts of it) shall be stored as a generic model is also an open issue.

# 6 Proposed Architecture of the System

We now want to discuss the proposed architecture from three perspectives. First, we illustrate the architecture from a point of view showing where specific kinds of knowledge reside. Then we discuss a user's view on the system. At last, we concentrate on the technical perspective.

## 6.1 Knowledge Structuring Perspective

Based on the ideas and requirements described above, we propose a three tier system architecture (see Figure 1). This architecture allows to distinguish between

- reusable process models,
- knowledge needed for a specific project, and
- knowledge & data created during project execution.

Knowledge is distributed over the three tiers. Within each tier, the *knowledge is structured in a process-oriented fashion*: Knowledge is linked to processes to be carried out in the course of the project. Instead of searching in the whole body of knowledge available in the system, the user

working on a specific task sees links to the knowledge associated with this task.

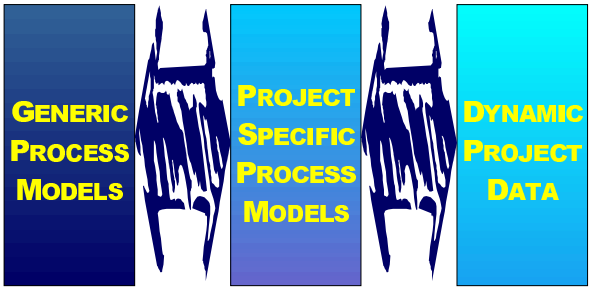



Figure 1: Three-tier architecture

### 6.1.1 Generic Process Models

The first tier handles generic and reusable process models and associates generic knowledge to the entities of the process model. Knowledge may be stored in several forms:

- Concrete knowledge chunks for human use, are stored URL references pointing to HTML pages or other files containing information in specific document formats. Access to this can be guaranteed using standard web mechanisms of binding file types to specific applications. Portability and cross-platform access is supported as long as the mapping of file types to applications are valid and the application software is available on the client platform.
- Formalized knowledge bases can be accessed and incorporated into the project execution in two ways. First, by downloading the KBS as a file and starting it on the client side. This requires the KBS interpreter on the client side and the association of the file type with the KBS interpreter application. Second, running KBS interpreter on the server side is an alternative. This requires the KBS to have a Web interface. A communication infrastructure between the KBS and the project enactment engine would even allow for transferring results from the KBS to the project execution. So, process-oriented knowledge structuring is a natural way for integrating KBS approaches into software development.
- Predefined queries are used when the knowledge chunk cannot be defined explicitly. The query describes the knowledge needed intentionally. We can at least distinguish between several kinds of queries:
    - Database queries: To access knowledge stored in a relational database, the process model has to contain SQL queries that return the appropriate result.
    - Information retrieval requests: Unstructured information can be accessed using information retrieval technology. The request may be posted to a special IR server or to one of the public web search engines.
    - CBR requests: Case-based reasoning technology integrates structured and unstructured queries. It allows for similarity-based queries on structured data whereas relational databases mainly support Boolean queries and IR systems work with unstructured data.

During project enactment, the access to generic process models and the associated knowledge is restricted to read only.

### 6.1.3 Project Specific Process Models (Project Plans)

The second tier contains project specific process descriptions. Using the single representation trick (known from machine learning), the mapping of generic process models to project specific process descriptions is easy: We use the same representation for both tiers and are able to copy generic models to a project. Then the generic descriptions are customized to be used in the specific project.

Copying models from the first tier to the second tier is easy. Open problems surely exist:

- which parts of a process model shall be copied (selection)
- how to determine that a generic model can be reused in a given situation
- how to support the customization and adaptation process

One of our students currently works on the first questions: Defining attributes for parts of process models that allow to select them in concrete projects using a similarity-based case retrieval algorithm.

Project plans contain the knowledge about the tasks to be done and the knowledge related to them. They are a basis for project enactment and coordination. Using a process enactment engine in a project, a project plan is the basis for *actively guiding* human users in their work.

### 6.1.4 Dynamic Project Data

The third tier handles dynamic knowledge which is the core of a flexible process engine[2]: The state of the work process and its tasks, do-do lists for its users, the products created during process enactment, causal relationships between process entities, etc.

The knowledge stored in this tier is created during process execution: it is the output of the work processes. In software development processes this includes e.g. requirements specifications, design documents, design rationales, traceability matrixes, source code etc.

The third tier provides - beside a product-oriented view - also a *process-oriented view* on the data created during task enactment. User are able to access information based on the processes carried out and they can follow the information flow in the project (thereby tracing where and by whom a specific information was used).

## 6.2 User scenarios

The users are working with the system from different perspectives and with different goals. The system allows to model and to access generic models and project plans. It provides to-do agendas for the users involved in the project. Furthermore the user can access background information of a project via an *information assistant*. In this section we give a short insight how users may work with the system.

Figure 2 shows the graphical interface for describing the information flow within a process model. The editors for defining the process decomposition and for editing product and resource

---

[2] In this paper, we do not discuss the design and implementation of our workflow engine and its unique features. Detailed information can be found at [DMP-97] or at the web site http://wwwagr.informatik.uni-kl.de/~comokit

models are omitted.

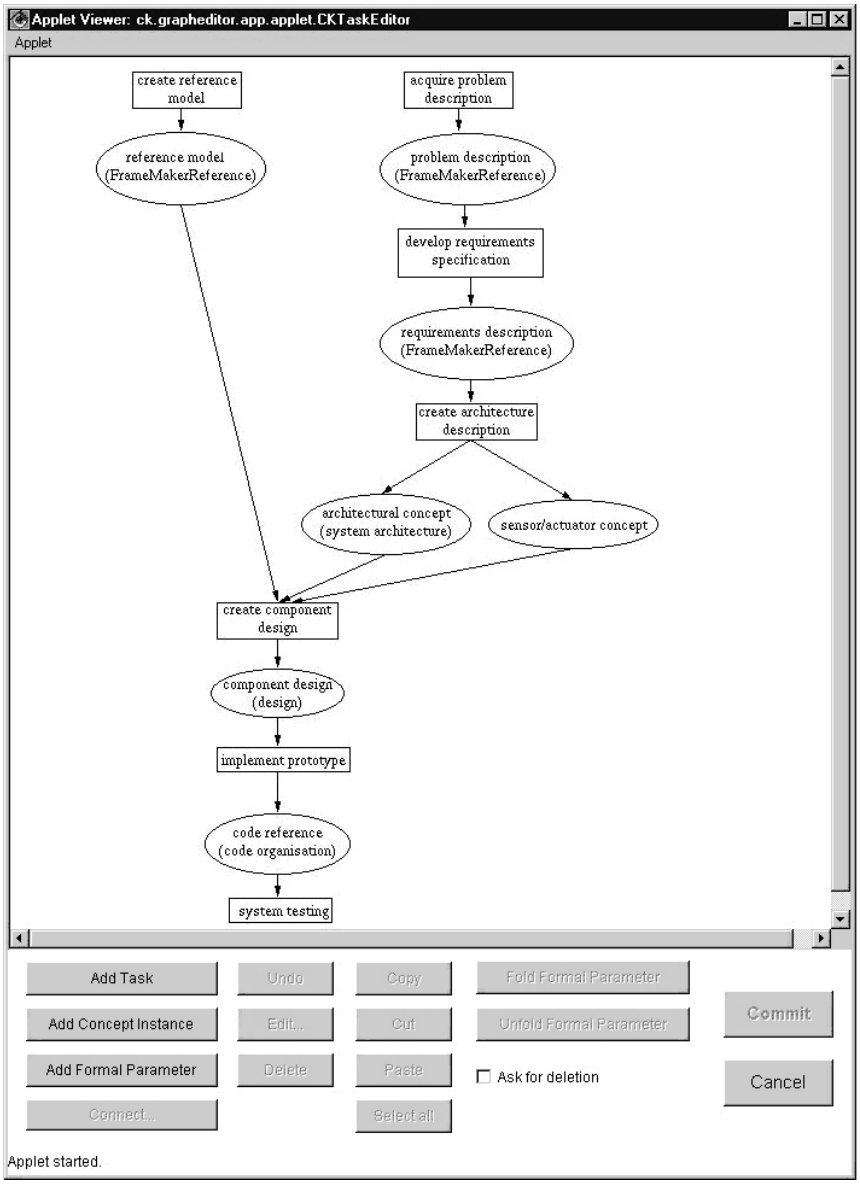
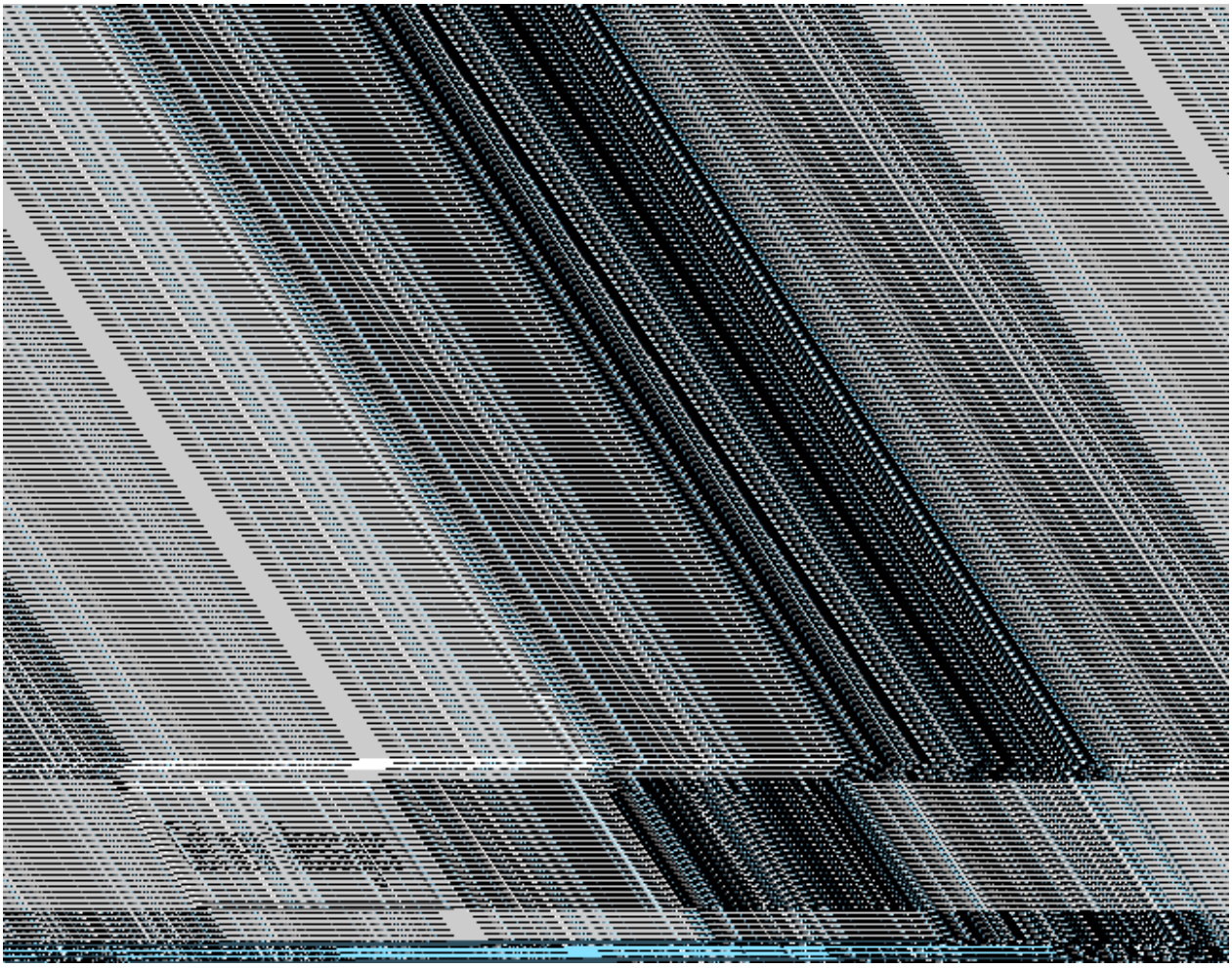


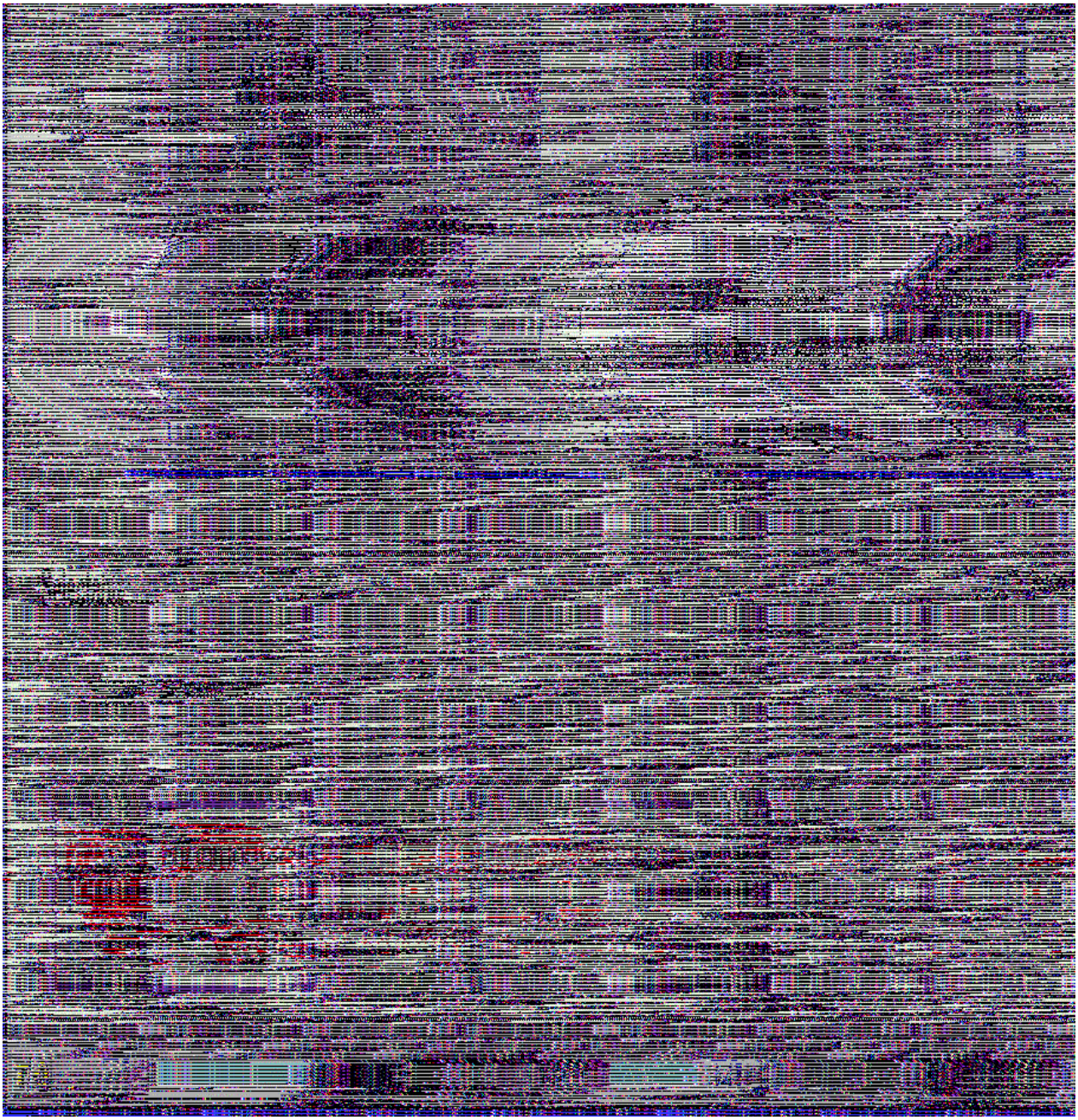

Figure 2: Information flow editor

The interface for defining project plans will look similar. In addition, the user will have access to the model library from where he can copy suited models.

For project execution, the system provides workspaces for managers, developers, and planners. The workspaces are very powerful, because they provide the user with relevant project data, guide them through their tasks, and automatically start required tools.

Figure 3 shows the workspace of *Mr. Holz*, who is actually executing an atomic method. His task is to develop an interface component in Java based on an OMT specification. The left window (A) shows him the documents he can access and he has to produce in order to solve the task. Clicking on the document symbols, customized editors to edit the products are automatically opened. In this example, a postscript viewer on the OMT specification (B) and an editor to edit the component (C) have been opened. After finishing the work, the task window and dependent tools will be closed by the system.





management and use http as the protocol to access versions of an object stored on arbitrary configuration management servers.

Storing information persistently, several technologies can be used: File systems, relational databases and object-oriented databases. File systems are restricted concerning version management, transaction support, and replication support. On the other hand they are easily accessible by (commercial) web servers. Databases support transactions and - often - replication of data. Relational databases only support table-oriented structures. More complex data structures are supported by object-oriented databases. Therefore, they are the means of choice for software process support (which is our main application area). Using OODBMS technology, we are implementing a configuration management system that allows to store arbitrary versions of products. We will extend this to support distributed configuration management using the built-in capabilities of the OODBMS GemStone (Requirement 5).

Resulting from the remarks above, we can define a first architecture for process support environments (Figure 6). The Internet is the backbone for the distributed system. Client computers access process data using Web browsers. Process models, project plans and dynamic project data are stored in a central object-oriented database with additional information stored in files. The database server supports project execution by implementing a process engine that handles to-do lists for team members and by providing access to all project data.

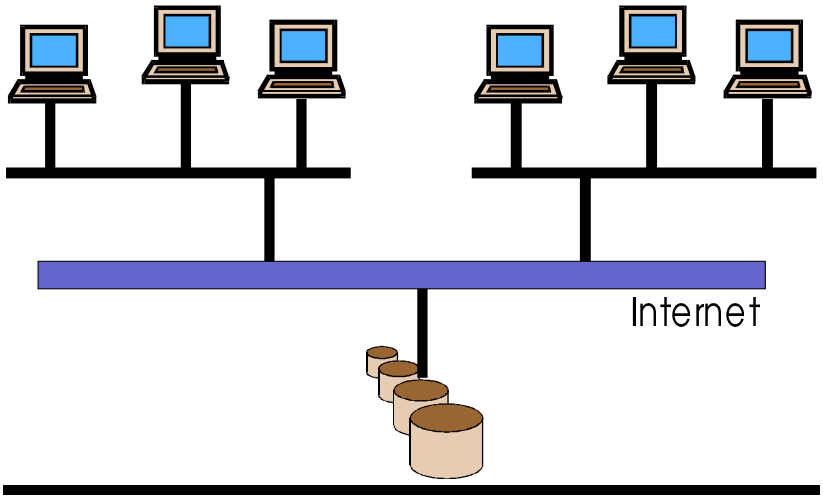



Figure 6: Centralized management of process knowledge

The problem with this architecture is that – due to the low bandwidth available on some Internet links – it does not always fulfil Requirement 4 (fast access to process knowledge) and Requirement 5 (distributed configuration management). These problems can be overcome by using proxies, caches and replication techniques that bring the information nearer to the end user. The process support system is then responsible for providing transparency by caching or replicating process data at different locations. Figure 7 shows an appropriate system architecture.

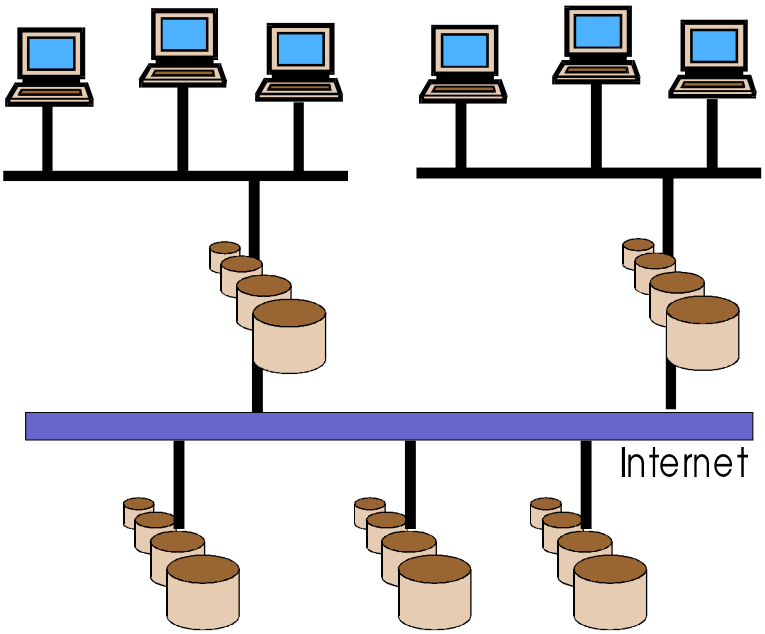



Figure 7: Cached/replicated & distributed management of process knowledge

The next question to be answered is how to implement this distributed architecture. Several frameworks for distributed system implementation were developed in the last couple of years. The main competitors in the market are now:

- OSF Distributed Computing Environment (DCE)
- DCOM from Microsoft
- OMG Common Object Request Broker Architecture (CORBA)
- Java Remote Method Invocation (Java RMI)

The DCE is based on a procedural approach and not state of the art anymore because of the wide spread use of object-oriented technology and implementation languages. DCOM is desktop-centered and dominated by a single company (Microsoft). In addition, it is currently restricted to Windows NT environments (failing to meet Requirements 3 and 4). CORBA is a mature standard developed by a consortium of more than 700 companies. It is platform and language independent (fulfilling Requirement 3 and 4). Several implementations are available since the early nineties – making it a comparable mature basis. Java RMI is rather new but gets a lot of interest from companies. It is platform independent (as far as Java is) but it is language dependent. In the future, Java RMI is expected to be built on top of CORBA. For our implementation, we decided to use a mixture of Java RMI and CORBA. CORBA allows building wrappers around legacy systems and integrating them into the process enactment environment. Java RMI supports the migration of objects over networks and therefore is a basis for implementing replication mechanisms.

We use Java applets to reduce the problem of distributing (project specific) software tools to team members whereas CORBA allows object-based communication between tools.

Requirements 2 (asynchronous work support) will be fulfilled using our published workflow management approach [DMP-97, Mau-97]. Our approach focuses on methods and techniques that increase the flexibility of workflow management by alternating project planning and project

execution steps (Requirement 7). In addition, we developed an approach that supports project coordination by automatically sending change notifications to appropriate team members. This functionality is based on traceability relations that are generated automatically using knowledge contained in process models [MP-94, DKM-96]. Our current approach to traceability is somehow limited: It only generates causal relationships between process data based on generic process notions (processes, process decomposition, and information flow). One of our students, Quan Li, currently develops a framework that allows to define domain-specific traceability relations using event-condition-action rules [ACT-96]. The extended framework fulfils Requirement 8 as soon as the notifications - which currently are only distributed in a local area network - are distributed over the Internet.

Product models specify the structure and relationships of (software development) products. In that sense, they describe the ontology of the domain - although they are not formally specified using logical representations (e.g. Ontolingua, KIF, or terminological logic). Using the restricted expressiveness of object-orientation, our system is neither able to automatically classify instances nor can it build up inheritance hierarchies automatically. On the other side, object-oriented approaches support the definition of arbitrary methods for objects. In our opinion, this is an advantage - from a system implementation point of view - compared to the use of terminological languages.

Requirement 1 (Synchronous work support) is not in the focus of our research. Therefore, we decided to use available technology for incorporating this functionality into our environment. Microsoft's NetMeeting and Netscape's Conference tools allow for audio and/or video conferencing over the Internet as well as shared whiteboards and/or shared applications (although this functionality currently is restricted to Windows PCs).

Figure 8 gives an overview over the architecture of the process knowledge manager.

The core component of the manager manages objects and provides (unique) names for the objects of the system. For every object, it handles multiple versions. It is able to return the current version of all objects stored. It also provides access to versions.

The project plan manager stores and provides access to all process-related information such as processes, process decompositions, resource assignments, etc. The component is central for handling all process-related information of the project plan and has to support the retrieval of the current plan information as well as old versions.

The dependency management component stores instantiated patterns of change dependencies of the project that are used to propagate and to notify affected agents about changes that occur during project execution.

The resource pool component basically manages a list of all team members (acting agents). For every team member, it maintains a calendar that stores on which days she is available for working in projects.

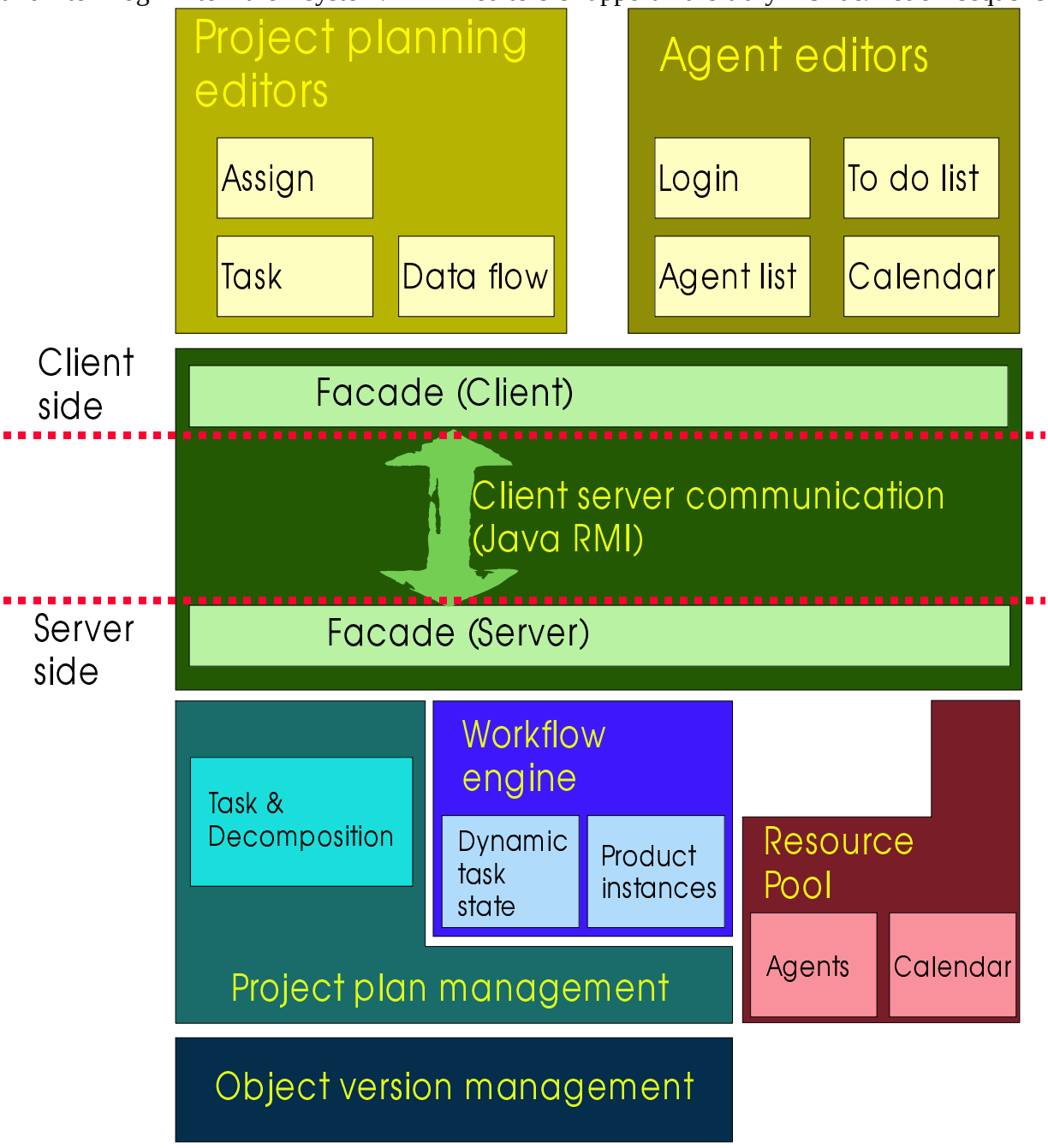
The workflow engine is a highly dynamic component and responsible for storing and accessing the current state of the project, e.g. the state of the tasks, the availability of inputs, references to the locked and finished outputs, and responsibilities for processes. The workflow engine closely interacts with the (static) project plan management component as well as with the resource pool.

The data on the server are accessed by a RMI interface. It provides a facade around the low-level details of the communication between the editors and the server objects.

The client side consists of a set of project planning editors, and editors that support the agent's work during project execution.

The project planning editor component is a graphical user interface that allows to edit project plans based on the modeling language MILOS. The plans are stored within the project plan management component.

The agent editor component is a graphical user interface that allows to edit agents and their properties (name, password, etc.), to edit the calendar of an agent to define when he is available to work on processes, to show the to do list of an agent and enter the result of process execution, and to login to the system. All editors support arbitrary Undo/Redo sequences.



# 7 State of Implementation

The whole idea of the paper is to present a concept for an Internet-based and process-centered knowledge management environment. Currently, the system is under development and not yet fully implemented. Nevertheless, our CoMo-Kit prototype implementation covers most of the future system's functionality (see Section 6.2) BUT is restricted to local area networks and supports only a restricted set of MILOS modeling primitives.

CoMo-Kit was developed using the OODBMS GemStone/S and Visualwave for Smalltalk as the programming environment. GemStone/S data definition language and data manipulation language is a Smalltalk dialect. Therefore, we are able to prototype systems in the Visualwave environment and than transfer part of the implementation into the GemStone database (gaining distribution over local area networks and transaction management).

The implemented prototype supports process modeling and process enactment. It contains a traceability component that generates causal relations based on process models and uses these to send change notifications to appropriate users.

We currently are in the process of re-designing and re-implementing all user interfaces as Java applets. An initial prototype that allows to model the information flow in processes will be available in February 1998. The re-implemented user interfaces are Internet-enabled and support the access to a centralized database that stores all project information.

In addition, a complete re-implementation of a MILOS-based process modeling and process enactment environment is undertaken as a joint effort of the groups of Dr. Richter and Dr. Rombach (both University of Kaiserslautern, Germany) and Dr. Maurer (University of Calgary, Canada).

# 8 Related Work

Software process modeling and process enactment approaches are discussed in the introduction to the paper. The same holds for workflow management. Here we will only discuss other work in knowledge management.

Work in knowledge management is influenced from several perspectives. On the one side, there is the organizational perspective that discusses how companies can organize their knowledge management [HSK-96]. Often these approaches concentrate on human resource aspects [SC-96].

[Sim-96, DGA-96] discuss the problem of acquiring the knowledge for knowledge management systems.

Knowledge management for distributed enterprises is discussed in [GNA-96]. The authors show how the web can be utilized for knowledge management. Their approach does not explicitly model the work process.

Several authors developed tools for managing different aspects and representation formalisms over the web [GS-96, FFR-96, Kre-96]. None of these is explicitly representing work processes (which - in our opinion - are the glue that holds several products together).

[ABH-96] integrates information retrieval and knowledge management. Although their approach also deals with work processes, they do not represent them explicitly so that the processes can be changed on the fly.

# 9 Summary

In this paper we described *requirements on* and *a concept for* an Internet-based process-centered knowledge management environment. The environment structures knowledge around work processes: In the center of our representation are processes. Linked to a process, the user can find methods (describing ways how to perform the process to reach its goals), products (input and outputs to the process), factual knowledge in the form product instances (often implemented as web references), and knowledge about the qualifications needed to perform the process.

The process-centered structure of the system has the following advantages:

- Processes are „natural“ entities for managers and team members: they are well used to thinking about the process (e.g. for project planning).
- For their daily work, people don't need knowledge per-se but the knowledge they need for performing specific task. A process-centered knowledge management system associates explicitly the task with the knowledge needed for it.
- By linking web references to task, the lost in hyperspace problem is reduced because the user immediately finds the knowledge needed instead of being forced to browse to relevant pages.

A process engine that guides the human users in their daily work interprets the explicit description of processes. This guidance is especially useful for new employees because they lack the knowledge about the standard procedures of a company.

# 10 Acknowledgements

We thank our colleagues Harald Holz, Boris Koetting, and Gerd Pews for fruitful discussion about the future redesign of our CoMo-Kit system. Sascha Schmitt currently is developing a prototypical interface of CoMo-Kit to the Web and testing the concepts described in this paper. We thank him for providing the screenshots of the Java applet supporting project planning over the Web.

# 11 References

[ABH-96] Andreas Abecker, Ansgar Bernardi, Knut Hinkelmann, Otto Kühn, Michael Sintek: Towards a Well-Founded Technology for Organizational Memories, AAAI Spring Symposium on Artificial Intelligence in Knowledge Management, AAAI Technical Report, Stanford, 1996

[ACT-96] ACT-NET Consortium. The Active Database Management System Manifesto: A Rulebase of ADBMS Features. ACM Sigmod Record 25(3): 40-49, 1996.

[AK-94] J. Armitage, M. Kellner. A conceptual schema for process definitions and models. In D. E. Perry, editor, Proceedings of the Third International Conference on the Software Process, p. 153–165. IEEE Computer Society Press, 1994.

[BFG-93a] S. Bandinelli, A. Fuggetta, S. Grigolli. Process Modeling-in-the-large with SLANG. In IEEE Proceedings of the 2nd International Conference on the Software Process, Berlin (Germany).

[BFG-93b] S. Bandinelli, A. Fuggetta, C. Ghezzi. Process Model Evolution in the SPADE Environment. IEEE Transactions on Software Engineering. Special Issue on Process Evolution, December 1993.

[BK-95] Douglas P. Bogia ad Simon M. Kaplan. Flexibility and Control for Dynamic Workflows in the wOrlds Environment. Proceedings of the Conference on Organizational Computing Systems, November 1995

[Bas-89] V. R. Basili, "The Experience Factory: packaging software experience," in Proceedings of the Fourteenth Annual Software Engineering Workshop, NASA Goddard Space Flight Center, Greenbelt MD 20771, 1989.

[BCR-94] V. R. Basili, Gianluigi Caldiera, and H. Dieter Rombach, "Experience Factory," in Encyclopedia of Software Engineering (John J. Marciniak, ed.), vol. 1, pp. 469--476, John Wiley Sons, 1994.

[BK-93] I. Ben-Shaul, G. Kaiser: Process Evolution in the Marvel Environment, 8th International Software Process Workshop : State of the Practice in Process Technology, March 1993.

[BA- 96] S. Bohner, R. Arnold: Software Change Impact Analysis, IEEE Computer Society Press, 1996

[BLR-95] A. Bröckers, C. Lott, H. Rombach, M. Verlage: MVP–L language report version 2. Technical Report 265/95, Department of Computer Science, University of Kaiserslautern, Germany, 1995.

[CHL-94] R. Conradi, M. Hagaseth, J.O. Larsen, M. Nguyen, G. Munch, P. Westby, W. Zhu: EPOS: Object-Oriented and Cooperative Process Modeling. In PROMOTER book: Anthony Finkelstein, Jeff Kramer and Bashar A. Nuseibeh (Eds.): Software Process Modeling and Technology, 1994, p. 33-70.Advanced Software Development Series, Research Studies Press Ltd. (John Wiley).

[CFF-93] R. Conradi, C. Fernström, A. Fuggetta. A conceptual framework for evolving software processes. ACM SIGSOFT Software Engineering Notes, 18(4): 26–35, October 1993.

[Con-97] R. Conradi: Cooperative Agents in Global Information Space, http://www.idi.ntnu.no/~cagis/

[CNG-95] G. Cugola, E. Di Nitto, C. Ghezzi, M. Mantione. How to deal with deviations during process model enactment. In Proceedings of the 17th Int. Conf. on Software Engineering (ICSE 17), 1995.

[CK-92] B. Curtis, M. Kellner, J. Over: Process modeling. Communications of the ACM, 35(9): 75–90, Sep 1992.

[DKM-96] B. Dellen, K. Kohler, F. Maurer: Integrating Software Process Models and Design Rationales, Proceedings Knowledge-Based Software Engineering KBSE-96, IEEE press, 1996.

[DMP-97] B. Dellen, F. Maurer, G. Pews: Knowledge-based techniques to increase the flexibility of workflow management, Data & Knowledge Engineering Journal, Vol. 23 No. 3, page 269-295, September 1997.

[DGA-96] Rose Dieng, Alain Giboin, Christelle Amergé, Olivier Corby, Sylvie Després, Laurence Alpay, Sofiane Labidi, Stéphane Lapalut: Building of a Corporate Memory for Traffic Accident Analysis, Proc. KAW-96, Banff, Canada, 1996.

[FFR-96] Adam Farquhar, Richard Fikes, James Rice: The Ontolingua Server: a Tool for Collaborative Ontology Construction, Proc. KAW-96, Banff, Canada, 1996.

[GNA-96] Brian R. Gaines, Douglas H. Norrie, Andrew Z. Lapsley, Mildred L.G. Shaw: Knowledge Management for Distributed Enterprises, Proc. KAW-96, Banff, Canada, 1996.

[GS-96] Brian R. Gaines, Mildred L. G. Shaw: A Networked, Open Architecture Knowledge Management System, Proc. KAW-96, Banff, Canada, 1996.

[GH-95] D. Georgakopolous, M. Hornick, An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure, in: Distributed and Parallel Databases 3, (1995) 119-153.

[HSK-96]G. van Heijst, R. van der Spek, E. Kruizinga, Organizing Corporate Memories, Proc. KAW-96, Banff, Canada, 1996.

[KFP-88]G.E. Kaiser P.H. Feiler, S.S. Popovich: Intelligent Assistance for Software Development and Maintenance (IEEE Software, May 1988).

[Kai-97] G. Kaiser: OzWEB, http://www.psl.cs.columbia.edu/ozweb.html, 1997.

[Kre-96] Rob Kremer: Toward a Multi-User, Programmable Web Concept Mapping "Shell" to Handle Multiple Formalisms, Proc. KAW-96, Banff, Canada, 1996.

[Mau-97] F. Maurer: CoMo-Kit: Knowledge Based Workflow Management, Proceedings Workshop on Knowledge Management, AAAI Spring Symposium, pages 106-110, 1997.

[MP-93] N. Madhavji, M. Penedo. Guest editor's introduction. IEEE Transactions on Software Engineering, 19(12):1125–1127, December 1993. Special Section on the Evolution of Software Processes.

[MP-94] F. Maurer, J. Paulokat: Operationalizing Conceptual Models Based on a Model of Dependencies, in: A. Cohn (Ed.): ECAI 94. 11th European Conference on Artificial Intelligence, pages 508-515, John Wiley & Sons, Ltd, 1994.

[Ost-87] L. Osterweil, Software Processes are Software Too, Proceedings of the Ninth International Conference of Software Engineering, Monterey CA, March 1987, pp. 2-13.

[PEM-95] G. Pérez, K. Emam, N. Madhavji: Customizing Software Process Models. In Proceedings of the 4th European Workshop on Software Process Technology, pp. 70 -78, Springer, 1995.

[PSW-92] B. Peuschel, W. Schäfer, S. Wolf: A Knowledge-based Software Development Environment Supporting Cooperative Work. In International Journal on Software Engineering and Knowledge Engineering, 2(1): 79-106,1992.

[RV-95] H.-D. Rombach, M. Verlage. Directions in software process research. In M. V. Zelkowitz, editor, Advances in Computers, vol.41, pages 1–63. Academic Press, 1995.

[SC-96] M. Sierhuis, W. J. Clancey: Knowledge, Practice, Activities and People, AAAI Spring Symposium on Artificial Intelligence in Knowledge Management, AAAI Technical Report, Stanford, 1996.

[Sim-96] Gaële Simon: Knowledge Acquisition and Modeling For Corporate Memory: Lessons Learnt from Experience, Proc. KAW-96, Banff, Canada, 1996.

[SOH-95] S. Sutton, L. Osterweil, D. Heimbigner: APPL/A: a language for software process programming, IEEE Transactions on SE and Methodology, Vol. 4, No. 3, p. 221-286, 1995.

[TKP-94] A. Tong, G. Kaiser, S. Popovich, A Flexible Rule-Chaining Engine for process Based Software Engineering, 9th Knowledge-Based Software Engineering Conference, September 1994

[Ver-94] M.Verlage, Multi-view modeling of software processes, in: B. C. Warboys, ed., Proc. Third European Workshop on Software Process Technology, (Springer Verlag, 1994) 123-127.

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

[WFMC-96] Workflow Management Coalition: Terminology & Glossary, http://www.aiai.ed.ac.uk/WfMC/DOCS/glossary/glossary.html