Internet Based Software Process Support

Frank Maurer
University of Calgary
maurer@cpsc.ucalgary.ca
sern.ucalgary.ca/~maurer

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

Abstract

The paper presents a process-oriented view on knowledge management in software development. We describe requirements on process support systems for globally distributed software development projects, introduce the process modeling language MILOS, and an enactment environment based on it.

1. 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 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 tasks, and of costs. This business objective leads to approaches as *lean management* and *business process reengineering & optimization*.

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 over time and budget
- time and cost estimations are error prone

Due to recent developments (e.g. the widespread use of Internet technology), the pressure on software engineers to deliver their products faster increases. Major problems in the resulting larger and distributed teams are

- how to coordinate and manage the distributed work of the software engineers
- how to manage the experience gathered in past projects and make them usable for the new ones

In order to fulfill these objectives, workflow management approaches often are introduced in the enterprise [15]. 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." [29].

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 has been addressing this problem.

Although workflow management has its limitations, it provides a process-oriented view on software development.

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 [22, 12, 1, 28]. 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 e.g. the Capability Maturity Model) as well as the actual basis for the coordination and the management of the software projects.

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 [25].

Software process modeling and enactment is one of the main areas in software engineering research. Several frameworks have been developed (e.g. procedural [26], rule-based [16, 27, 24], Petri net based [2], object-oriented [8]).

Process-sensitive software engineering environments which support evolution of executed process models [9] are a focal point of current research, but the results are still immature [19]. Several approaches to support the evolution and flexibility of software processes were developed [3, 4, 11, 23]. None of these approaches is supporting globally distributed software processes and their evolution (although work in this direction is starting [17, 10]).

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 [7] .

Managing software process knowledge is also the goal of the experience factory approach [5, 6]. 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.

Our aim, and the subject of this paper is to integrate workflow management, and knowledge based system approaches, to gain a process-oriented view on knowledge-intensive tasks in software development. The resulting approach shall in particular meet postulated requirements on flexibility, and distributed work support (section 2). To model knowledge intensive workflows, we provide the process modeling language MILOS (section 3). Section 4 describes the architecture of the resulting software process support system. An overview over the state of implementation is given in section 5. The last section is a summary.

2. 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 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, and different hard- and software requirements

Although many of 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. A more comprehensive discussion can be found in [21].

Requirement 1: Asynchronous work support: In globally distributed software development processes, people work asynchronously in different locations and at different times. A support system shall facilitate the coordination and the management of this process.

Requirement 2: Flexible project plans: 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 while the development is already in progress.

Requirement 3: 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).

3. MILOS: Process Modeling & Planning

To integrate process modeling and knowledge management techniques, we developed the process modeling language MILOS in cooperation with the working group of Prof. Rombach.

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 (manuals, standards).

Below we shortly describe MILOS basic language concepts for defining process models.

3.1. 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.

3.2. Process Models

Within process models activities and their interrelationship are described.

A process is defined by a description of the process goal, the products needed 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 several subprocesses whereas an atomic method describes an atomic way to reach the process goal without refining the process.

Inputs of a process may either be products, that are produced by other processes during project enactment, or organizational knowledge, such as guidelines, templates, experience reports, etc.

Each 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”).

3.3. Resource Models

Resource models allow to assign roles and qualifications to project team members. These models describe knowledge needed by project managers to find appropriate people for a task.

3.4. Project Plans: Customized Process Models

Process models are generic, reusable descriptions of how to execute projects and are part of the organizational knowledge. 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 may be composed of several process models. For instance, the test processes of a project plan has been adopted from a model for testing, whereas the general course of action has been adopted 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 that meets the company’s quality standards and procedures.

4. Structure of the System

In this section we will discuss the system’s architecture from two perspectives. We illustrate the architecture showing where specific kinds of knowledge reside. Then we focus on the technical perspective.

4.1. Knowledge Structuring Perspective

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

- reusable/generic process models,
- project specific process models (project plans), and
- data created during project execution.

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.

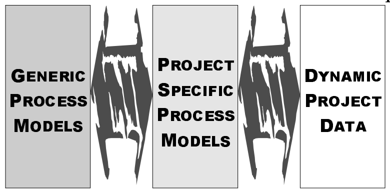

Figure 1: Three-tier architecture

4.1.1. 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 via URLs pointing to HTML pages or other files containing information in specific document formats (with helper applications or plug ins configured appropriately). Examples are company process standards, manuals etc.
- Predefined queries are used when the knowledge chunk can not be defined explicitly. The query describes the knowledge needed intentionally.

4.1.2. 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.

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.

4.1.3. Dynamic Project Data

The third tier handles dynamic data which is the core of a flexible process engine (more details can be found at [14]): 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 information 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 also provides - beside a product-oriented view - a *process-oriented view:* Users are able to access information based on the information flow in the project (thereby tracing by whom a specific information was created or used).

4.2. Technical Perspective

This section discusses design alternatives for building the system and explains why we use a specific technology.

To use a ubiquitous communication infrastructure, Internet technology is clearly the way to go. The Internet - and, in the future, the Internet II - will provide this infrastructure to lesser costs than point to point (satellite) lines. In fact, the wide spread use of Internet-based communication as well as the increasing speed of its introduction to a “standard” working environment make it the obvious and cost-effective choice for global teams. Internet-based virtual private networks create a secure environment for work groups based on encryption techniques. Current problems, e.g. bandwidth, latency, quality of service guarantees, will be overcome with the broad introduction of Internet II and the new protocols. Internet access is more or less available in every workplace - or will be in the next couple of years. To be able to access process and/or project information, the only widely available cross-platform environment is the web. Other environments are either restricted to one platform or use proprietary communication protocols that are not supported in every company. To be able to build up virtual corporations, we decided for a common denominator: TCP/IP and Web browsers.

For 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. These also allow for executing code in the server. Therefore, they are the means of choice for software process support. Resulting from the remarks above, we can define a first architecture for process support environments. 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.

For implementing this distributed architecture we use the OMG Common Object Request Broker Architecture (CORBA) and Java Remote Method Invocation (Java RMI)

CORBA is platform and language independent. Several implementations are available since the early nineties – making it a comparable mature basis.

Java RMI 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.

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.

Asynchronous work is supported by our workflow management approach [14, 18]. Our approach focuses on methods and techniques to support (a) plan refinement (b) plan adaptation, and (c) error correction during project execution (Requirement 2) [14]. To analyze, and manage the impact of a change on the project our system provides mechanisms for

- identifying affected processes, results (products) and resources, and, as far as possible, updating their state automatically,
- automatic notification of affected team members and managers about the change and its consequences (Requirement 3),
- tracing causal relations automatically generated using knowledge contained in process models [20, 13].

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). It will be extended through domain-specific traceability relations.

Figure 2 gives an overview over the architecture of the Internet-based process support environment MILOS. We currently are not implementing the Generic Process tier (cf. Figure 1). Therefore, the architecture shows only non-generic parts.

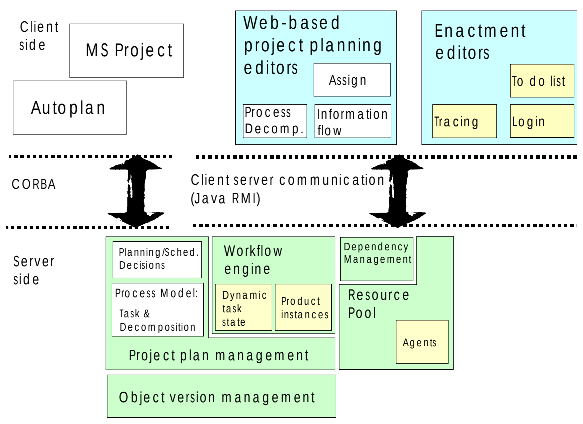

Figure 2: MILOS process support environment

The *object version management* stores objects and provides (unique) names for these. For each object, it handles multiple versions. It is able to return the current version of all objects stored. It also provides access to older versions.

The *project plan manager* stores and provides access to all process-related information such as processes, process decompositions etc.

The *dependency management* component stores instantiated patterns of change dependencies of the project that are used to determine the effects of changes and to notify affected team members.

The *resource pool* component basically manages a list of all team members. For each team member, it stores its qualifications and roles in the company.

The *workflow engine* is 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 Web-based project planning editor and enactment editor provide user interfaces for the server side components.

The COTS tools *Autoplan and MS Project* are used for scheduling and as a replacement for the web-based project planning editors.

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

5. State of Implementation

The MILOS system is based on ideas and a prototype implementation (in Smalltalk) developed within the CoMo-Kit project.

Currently, a complete redesign and re-implementation of the MILOS process modeling and process enactment environment is undertaken as a joint effort of the groups of Dr. Maurer (University of Calgary, Canada) and Dr. Richter.

The system is implemented in Java using the OODBMS GemStone/Java for the server side and applets for the client side.

A beta release (which will be made available on the Web site of the Canadian partner) is planned for September 1998. This release will be used by a multi-national company starting 1999.

6. Summary

In this paper we described *requirements on* and the *architecture of* an Internet-based software process support 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 system associates tasks with the knowledge needed for them.

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 knowledge about standard procedures of a company.

Acknowledgements

We thank our research assistants Dale Couprie, Mike Gao, Quan Li, Yunbo Wang, and Xuejing Yin (all U of Calgary) for their implementation effort.

The research presented in this paper was partially funded by Nortel, the University of Calgary, the National Sciences and

Engineering Research Council of Canada NSERC, and the German Research Foundation DFG.

References

[1] 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. IEEE Computer Society Press, 1994.

[2] 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.

[3] 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.

[4] 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

[5] 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.

[6] V. R. Basili, G. Caldiera, H. D. Rombach, "Experience Factory," in Encyclopedia of Software Engineering (John J. Marciniak, ed.), vol. 1, John Wiley Sons, 1994.

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

[8] 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. Advanced Software Development Series, Research Studies Press Ltd. (John Wiley).

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

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

[11] G. Cugola, E. Di Nitto, C. Ghezzi, M. Mantione. How to deal with deviations during process model enactment. In Proc. 17th Int. Conf. on Software Engineering, 1995.

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

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

[14] 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.

[15] 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.

[16] G.E. Kaiser P.H. Feiler, S.S. Popovich: Intelligent Assistance for Software Development and Maintenance IEEE Software, May 1988.

[17] G. Kaiser: OzWEB, www.psl.cs.columbia.edu/ozweb, 1997.

[18] F. Maurer: CoMo-Kit: Knowledge Based Workflow Management, Proceedings Workshop on Knowledge Management, AAAI Spring Symposium, 1997.

[19] N. Madhavji, M. Penedo. Guest editor’s introduction. IEEE Transactions on Software Engineering, 19(12): Dec 1993.

[20] 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.

[21] F. Maurer, B. Dellen: An Internet-Based Software Process Management Environment, Proceedings of the ICSE-98 WS on “Software Engineering over the Internet”, sern.ucalgary.ca/~maurer/ICSE98WS/ICSE98WS.html

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

[23] G. Pérez, K. Emam, N. Madhavji: Customizing Software Process Models. In Proc.4th European Workshop on Software Process Technology, Springer, 1995.

[24] 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), 1992.

[25] 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.

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

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

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

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