Coordinating Management Activities in Distributed Software Development Projects*

Fawsy Bendeck, Sigrid Goldmann, Harald Holz, Boris Kötting[1]
AG Expert Systems, University of Kaiserslautern, Postfach 3049, 67653 Kaiserslautern, Germany
e-mail: {bendeck, sigig, holz, koetting}@informatik.uni-kl.de

1. in alphabetical order

Abstract

Coordinating distributed processes, especially engineering and software design processes, has been a research topic for some time now. Several approaches have been published that aim at coordinating large projects in general, and large software development processes in specific. However, most of these approaches focus on the technical part of the design process and omit management activities like planning and scheduling the project, or monitoring it during execution.

In this paper, we focus on coordinating the management activities that accompany the technical software design process. We state the requirements for a Software Engineering Environment (SEE) accommodating management, and we describe a possible architecture for such an SEE.

1 Introduction

Large-scale, systematic software development is a process where different team members with different skills perform different activities. Usually, a distinction is made between technical-oriented and management-oriented activities, the former denoting activities that are part of the technical process (e.g. designing, coding and testing), while the latter denotes activities that are concerned with the administration and control of the technical activities.

[15] identifies several different technical (e.g. Requirements Engineer, Designer, Coder) and management roles (e.g. Project Planner, Project Manager, Quality Assurer) that should be present in systematic development processes. Each role is responsible for performing a specific set of activities and can be performed by more than one project team member, just as it is possible for a team member to perform more than one role.

Since communication becomes both increasingly difficult and more important as project size increases, coordination mechanisms are needed for large projects, especially if they are distributed geographically as well as logically. In the past, the coordination of technical roles in the software development process has been a research issue, but the integration and coordination of management roles has been mostly neglected. Here, we concentrate on coordinating the management activities in a large-scale project. Our goal is to coordinate the whole software development process (i.e. the management activities as well as the technical process), providing a notification mechanism that keeps all team members up to date on the current project state. We try to provide everybody with all information necessary to fulfill their tasks, thereby making geographical distribution as transparent to the team members as possible. Though global project distribution creates additional problems that we do not address here (e.g. data security, time difference, etc.), and which need further consideration, we nonetheless believe that our proposed system will solve the most pressing problems posed by distributed project coordination.

In section 2, we state the general requirements that an environment for distributed software development and management, i.e. a management-oriented Software Engineering Environment (SEE), should meet, as well as the requirements for such an SEE from the perspective of the management roles. Section 3 describes scenarios that

*This work is supported by the *Deutsche Forschungsgemeinschaft, Sonderforschungsbereich 501*

might occur during the software process, which underscore the necessity to integrate management roles into project coordination. Section 4 introduces an architecture proposal, and maps this architecture to the requirements stated in section 2. Section 5 describes the existing systems that our implementation will build on and extend. Finally, section 6 summarizes related research projects.

2 Requirements for a management-oriented SEE

Below, a list of requirements is presented that we believe an SEE must meet in order to support the coordination of management activities in software development projects. General requirements have been derived from activities that frequently have to be performed during project management. Role-specific requirements have been obtained by focussing on the different management roles introduced by [15] individually.

2.1 General Requirements

(1) *Process modeling and planning support*
Basic concepts of software process modeling like products, processes, product flow, control flow, resources, attributes and roles as well as quality models should be provided. They serve as a means to represent general, company-specific knowledge about processes, products and quality standards. This knowledge will have to be refined and instantiated in order to create a project-specific plan.

(2) *Scheduling support*
During scheduling the plan has to be augmented by time and resource allocations in accordance to estimations on time, cost and resources in the plan. Information like resource workloads and critical paths should be provided to support resource and time levelling.

(3) *Monitoring support*
During plan enactment, the SEE should provide appropriate views for different management roles to allow for monitoring the project's progress. Management activities require information about individual product, process and resource states as well as global information on the project like the number of processes finished or products completed. Other aspects of interest are cost information and quality values gained by measurement activities.

(4) *Evaluation support*
Project enactment must be evaluated to identify deviations from the project's schedule, cost or desired quality. Deviations should be classified in order to help the manager to focus on the most important ones. As far as possible, support should be given to identify the source of the deviations by analysing event/activity chains in the project history.

(5) *Support for the selection of corrective actions*
Evaluation only is concerned with the identification of deviations. Further support is needed to determine the right corrective actions to bring the project back in track. Change impact analysis techniques should be available to assist managers in the selection of changes to be made to the project's plan or schedule.

(6) *Replanning and rescheduling support during enactment*
Changes selected must be applied to the plan and/or schedule while it is being enacted. Replanning and rescheduling should be supported by notifying all team members concerned by the change (e.g. new processes require additional scheduling and measurement planning; deleting or modifying a process currently in execution requires informing the team members responsible for its scheduling and execution, etc.).

(7) *Interleavable planning, management, and enactment*
The development of complex products cannot in advance be planned to a level of detail that requires no further process refinements. In practice, planning, scheduling and enactment have to be interleaved. During enactment of a high-level plan according to schedule, sub-plans will have to be created for individual processes. Depending on the level of detail, these sub-plans will also require measurement planning, scheduling and other management activities of which necessity the responsible team members have to be informed.
Furthermore, the resources and estimated duration in a sub-plan for a process might violate the process's resource restrictions, estimated duration or scheduling. If this violation can not be solved by changing the sub-plan, the team member responsible for the high-level plan or schedule has to be notified of the problem.

(8) *Notification support*
The above requirements already mentioned the necessity to coordinate management activities and to inform other managers and technical agents about changes. Hence, the concept of to-do lists as well as the notification system that forms the basis for coordinating technical activities has to be extended to include management. A sophisticated notification system for sending and receiving general process coordination messages as well as change notifications between both management and technical roles must be provided by the SEE.

(9) *Dependency and change management*
Local changes to project information (e.g. the plan or documents that have already been produced) frequently occur during software development projects. To prevent

inconsistencies, changes must be propagated in order to identify all project information effected by a local change. Therefore, dependencies between activities must be maintained to allow for an automatic recovery of a consistent project state; furthermore, the dependencies have to be analysed in order to identify team members that must be notified in case of changes. In addition, dependency management serves as a basis for change impact analysis (see requirement (5)).

(10) *Include legacy software*
Project information might be stored in legacy software tools, and agents might have preferred tools that they should be allowed to work with (e.g. MSWord, MS-Project, AutoPLAN, etc.). The first point raises the requirement to at least view data stored in legacy tools, while from the second point, the requirement emerges for communication between legacy tools and our SEE.

2.2 Role-specific Requirements

Project Planner (PP)

The Project Planner's task is to create a project plan to reach the project goals (e.g. to produce a product with specified quality attributes inside a time scope or deadline prescribed by a contract), while heeding the project characteristics, such as available staff, or company-wide rules about best practice processes, general quality goals, etc.

In order to create a project plan, the PP accesses the experience base, and reuses product and process models as well as quality models[1] yielded by former projects.

Since software projects tend to deviate from the plan on a regular basis, the PP also needs to be able to change the plan during project execution whenever necessary. The Project Planner needs to be notified when changes to the project plan become necessary. On the other hand, changes in the plan must be propagated to the team members concerned by the changes.

Another service the system should provide for the Project Planner is statical and dynamical project plan analysis, e.g. consistency checks and simulations.

Measurement Planner (MP)

The planning of measurements to be taken during project execution is usually regarded as part of the Project Planner's responsibilities. Here, we distinguish between general planning activities and measurement planning, because the latter is an important activity by itself, and this distinction frequently occurs in Software Engineering practice.

The Measurement Planner should take a goal-oriented approach [1] at measuring process, product and resource attributes, and can specify desired values for those attributes. In cooperation with the Quality Assurer (see below), the MP builds a measurement plan to be enacted and monitored by the Quality Assurer during project execution. Since the measurement plan depends on the project plan, the Measurement Planner needs to be notified when the plan changes. In turn, the Measurement Planner must be allowed to change the project plan in order to add measurement activities, and to prepare the measurements, e.g. by creating questionnaires to be filled in by technical agents, and by specifying measurement tools which will take the required measurements during project execution.

If changes in the project goals come up during project execution (e.g. through contract alterations or changes in the company's quality goals), the measurement plan will need to be adapted, which in turn might make project plan changes necessary. The resulting communication between the Measurement Planner, the Project Planner and maybe the Project Manager should be supported by a management-oriented SEE.

Additionally, the SEE should be able to trigger the specified measurement tools, so that measurements can be taken automatically, and questionnaires be presented for technical agents to fill in.

Project Manager (PM)

The Project Manager ensures the correct and timely execution of the project. His/her responsibilities are to assign agents and other resources to the processes prescribed in the plan, determine the processes' start and end times in accordance with milestones and deadlines dictated by the plan (thereby creating a schedule for the project), and to watch the project plan execution to ensure that the project stays on time and meets other quality requirements described in the plan, like for example compliance to best practice processes. In order to rectify deviations from these requirements, the PM needs to be able to take corrective actions concerning the schedule. If changes to the project plan become necessary, the Project Manager will cooperate with the Project Planner to initiate these changes.

Our SEE needs to implement a notification mechanism which will inform agents assigned to processes of this assignment, and give them an opportunity to negotiate it. In order to point out deviations from prescribed timing or quality attributes to the PM, the system needs to monitor the process timing as well as specified attribute values and notify the PM in case of deviations from the desired values.

The Quality Assurer (QA)

The Quality Assurer is responsible for the compliance of process and product quality to the quality goals dictated by the contract and by company policy. In order to control this compliance, the QA takes part in creating the measurement plan, and during project execution, monitors the quality at-

1. A quality model contains information about desired values for quality attributes, which has been gained by experience in past projects.

tributes introduced by the measurement plan. If these attributes deviate from the desired values, or if the project history suggests that they might deviate in the future, the QA will trigger corrective actions by notifying the Project Manager, and if necessary, the Project Planner. These three will then debate the best course of action to take in order to meet the project's quality requirements.

In order to fulfill these responsibility, the QA needs access to all measured data (provided by measurement tools), and should also be provided with a monitoring service that notifies him/her if desired values are not met.

The Product and Quality Managers

These two roles in the software development process are responsible for the data in the experience factory (for a description of the two parts of an experience factory, see [2]): The Product Manager controls the versioning and configuration of product data produced during project execution and stored in a project specific part of the experience factory, while the Quality Manager is responsible for generalizing the experience from the company's projects, and packages it for reuse in later projects. These two roles have no direct influence on the project course and therefore will not be discussed further in this paper.

The Technical Roles in the Software Process

The technical roles are responsible for the execution of the processes described in the project plan. Technical roles are for example the Requirements Engineer, the Designer, the Programmer, the Tester, etc.

Coordinating these roles during project execution is an issue that has been much discussed, and is still not completely solved. In this paper, however, we will concentrate on the management-oriented roles, and omit the requirements that arise from coordinating the technical roles in an SEE. (See for example [5], [6] for a discussion of these requirements).

3 Scenarios for Distributed, Cooperative Software Development

In this section, we outline the management activities that need to be handled during a software development project. Assume a company with branch offices all over a country. The software development and its management can be distributed to these offices. Based on the roles described in section 2.2, we will now describe a number of scenarios for each of these roles.

3.1 Scenario from the Perspective of a Project Planner

Due to a contract change, the Project Planner receives a request for a change in the quality requirements: the reliability of the whole software product must be less than 1 fault (former 3 faults) in the final acceptance test. Here are some possible activities the PP might perform in order to increase the reliability of the software product:

- Contact the Quality Manager in order to request a new quality model concerning the relation of fault rate and effort distribution over the project stages.
- insert another test iteration into the project plan.
- make a new time estimation and increase the quantitative value of the quality attribute *calendar time* for the whole software development process from ten months to twelve months.

The concerned employees must be notified about all changes.

3.2 Scenarios from the Perspective of a Measurement Planner

During project execution, it might turn out that the planned course of action is not practical, and the plan has to be changed: processes will be deleted, and others added. In this case, the Measurement Planner has to be notified, so that the measurement plan can be adjusted accordingly. In a next step, the MP will add attributes to the new project plan, and insert corresponding measurement activities into the plan. The MP will then assign tools to these activities and/or specify questionnaires which at execution time will be presented to the responsible agents to fill in.

3.3 Scenario from the Perspective of a Project Manager

(1) An agent has to finish a process up to a given deadline. The system has already sent a message to the agent to remind him/her of this process. If the process has not been finished by the assigned end time, the Project Manager receives a corresponding notification from the system. The Project Manager contacts the agent to clarify the problem. Depending on the seriousness of the problem he/she needs to react accordingly. The agent is not able to do his/her work in the estimated time which can have multiple reasons, for example:

- the assigned end time was calculated too restrictively and has to be recalculated
- the agent's skill level is not as high as defined in the model, and the model has to be adapted
- the agent has to perform other process with higher priority so he/she cannot work on the concerning process. In this case, the agent's workload has to be reduced, or the process has to be rescheduled for a later time.

If other processes depend on the delayed process' output, the delay must be propagated to subsequent processes which will result in a correction of their start and end times. If there are fix delivery milestones, and the delay caused by the process in question would violate them, some other processes must be replanned and/or rescheduled in order to make up for the lost time. If this is not possible, the company's management must be informed about the delay, and the contract must be renegotiated.

(2) The Project Manager gets a notification that an agent is on sick leave. He/she needs to assign a new agent for the process. This decision should be supported by suggesting agents for a new assignment, and explaining this suggestion; possible explanations are:

- suggested agent is in the same development team
- suggested agent has nearly the same skills
- suggested agent is declared as the sick agent's stand-in
- suggested agent has low workload.

The Project Manager changes the schedule and the selected agent is notified.

(3) A process, which was scheduled to be finished on Wednesday, was finished on time. On Thursday, the responsible agent discovers an error and decides to re-execute it. In addition to informing the agents responsible for subsequent processes that their inputs are no longer valid, the responsible Project Manager has to be notified that the corresponding scheduling decision is now invalid, while the Project Planner needs to replan, and introduce a new copy of the process into the plan.

3.4 Scenario from the Perspective of a Quality Assurer

(1) The Project Planner has changed a quality model in the project plan, and the Measurement planner has adjusted the measurement plan accordingly. The Quality Assurer needs to be informed about the changes, which might include:

- new measuring methods and/or tools for some quality attributes
- new quality defaults and new desired values
- new attributes plus their handling and desired values.

(2) The Quality Assurer watches the progression of the robustness of a certain software component. The desired value for the component's robustness (according to the measurement plan) is less than two errors in the last component test. There are still five days to go until this desired value has to be reached, and a quality model from the Experience Base suggests that a component should have reached a robustness of less than seven errors at this development stage in order to be able to meet the desired value of less than two error at the last component test. Hence, the QA informs the Project Manager to take a look at this attribute. The Project Manager can apply to the responsible technical agent to find reasons for the problem and to look for a solution, which in turn might lead to plan changes.

4 Approach

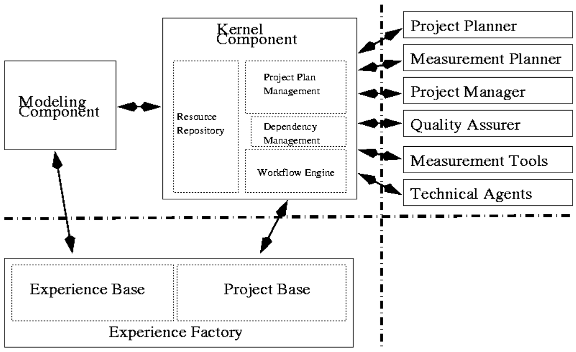

Figure 1

Figure 1 outlines the architecture we propose for a system that handles the above scenarios. We will give a short description of the system components, and then show how this architecture undertakes to meet our requirements.

The Modeling Component

The Modeling Component is a store for process, product, and quality models which have been retrieved from the Experience Base (EB) and then customized for the specific project. The Modeling Component provides several methods to work on each modeled process. The Project Planner or a technical agent can select one of these methods and thereby build and/or refine the plan.

This component also provides statical and dynamical analysis mechanisms to evaluate the model and possible plans derived from the model.

The Kernel Component

The Kernel Component maintains the plan and schedule as well as the project's execution state. It tracks dependencies between planning, scheduling and execution activities, and implements a notification system that keeps the concerned agents up to date on changes.

The Kernel Component is composed of several sub-components:

- The *Resource Repository* provides a complex resource system that represents roles, properties and skills of agents.

- The *Project Plan Management* handles the project plan and schedule.
- The *Workflow Engine* contains the project trace, manages plan execution, and handles the measurements specified in the measurement plan.
- The *Dependency Management* administers the connections between the planning and scheduling activities managed by the Project Plan Management component, and the execution and measurement activities handled by the Workflow Engine. It also implements the notification mechanism mentioned above.

A more detailed description of these components can be found in [6].

The Experience Factory

The Experience Factory has a structure as described in [2]. It stores general models (in the Experience Base) as well as specific project data (in the Project Base), for which it provides a suitable version and configuration management.

The Software Process Agents

The Project Planner, Project Manager, Quality Assurer, and Measurement Planner, as well as the technical agents, perform the activities described in section 2.2.

The actions of the above agents as well as the dependencies between these actions and other decisions made during project planing and execution, are communicated to and tracked by the Kernel Component.

The measurement tools are triggered automatically by the Kernel Component to take the measurements defined by the Measurement Planner. Depending on the kind of measurement to be taken, a tool will either pop up a questionnaire for a technical agent to fill in, or if possible, directly measure the needed value.

The Software Process Agents and the Kernel Component communicate over a defined protocol which can be implemented via a remote message call interface, socket communication, and/or the WWW. For an example implementation of an approach based on socket communication see [9].

4.1 Consistency of the Proposed Architecture With the Requirements)

In this section, we summarize the functionality our proposed architecture provides to meet the requirements stated in section 2.

(1) *Process modeling and planning support*

The Modeling Component stores the process and product models to be used to build up the plan, as well as resource models for scheduling decisions, and quality models that the project execution should comply with. The Project Planner as well as technical agents can use and/or enhance this model in order to plan the project or single tasks. The Measurement Planner is also able to access the model and plan in order to insert attributes into process, product and resource models, and plan the measurement activities to be performed during project execution.

(2) *Scheduling support*

The Project Plan Management in our architecture allows the Project Manager (and if necessary, technical agents) to assign processes to agents, and determine start and end times for them. The Dependency Management tracks the dependencies between these scheduling decisions and other activities during project planning and execution, and notifies all concerned agents in case of occurring changes. The system also provides role specific views on project information. For example, the Resource Repository supplies the Project Manager with information like agent workload and agent skills, and the Project Plan Management contains critical path information, in order to facilitate project scheduling.

(3) *Monitoring support*

In addition to the views provided by the Project Plan Management and Resource Repository (as mentioned above), the Workflow Engine provides information about the project's current state, i.e. about process and product states, and about measurement values taken during process execution.

(4) *Project evaluation support*

A sub-component of the Dependency Management will check the compliance of measured values (provided by measurement tools) with the corresponding desired values, and notify the Project Manager and/or the Quality Assurer, as well as other concerned agents, if deviations occur. If the project trace yields an explanation for the deviation (e.g. if a process is started late because of a delay in a preceding process), the Dependency Management component will deduct this reason and present it to interested agents.

(5) *Support for the selection of corrective actions*

The Dependency Management will provide change impact analysis support. For example, if a process has to be redone, the responsible agent (i.e. the Project Planner or a technical agent) can query the system for an estimation of the overall time and cost necessary to redo not only this process, but all dependent ones as well. In addition to this domain independent change impact analysis, we are currently discussing how to model domain-specific dependencies, in order to provide a more detailed change impact analysis [7].

(6) *Replanning and rescheduling support*

The system (i.e. the Dependency Management component) supports plan and schedule changes during project execution by tracking dependencies between plan and schedule on the one hand, and execution activities on the other hand. Changes are automatically propagated through the plan and schedule, and the concerned agents are notified.

(7) *Interleavable planning, management, and enacting*
Our Dependency Management component allows planning, scheduling, and execution activities to be alternated as described in [9]. As mentioned above, it tracks the dependencies between these activities and provides agent notification when information becomes available that is necessary to further plan, schedule, or execute a process or subproject.

(8) *Notification system*
As mentioned above, the notification system will be part of the Dependency Management component. The Workflow Engine provides to-do-lists for technical and management oriented agents.

(9) *Dependency and change management*
This Dependency Management not only supports dependency tracking between management activities, but manages the dependencies between technical activities and between technical and management activities as well, providing notifications in case of changes. It also provides support for backtracking to a consistent project state after a change has occurred.

(10) *Legacy Software*
Each of the agents on the right side of figure 1 might be a legacy tool. The technical issues of how to integrate legacy software are not trivial, but out of the scope of this paper. Another problem we are currently discussing is that legacy tools often do not share our view on the different roles. For example, common project management tools (like MSProject or AutoPLAN) do not distinguish between planning and scheduling activities. In order to integrate such tools with our system, we will implement wrappers for the tools which "translate" the tools' view on the project into terms that our Dependency Management can track. This point needs further research and has to be discussed individually for each legacy tool to be integrated in our system.

5 State-of-implementation

Our experience and ideas are based on different existing systems that we will integrate in order to build the suggested architecture:

The base component for dependency management is Redux [14]. Its primary purpose is to provide a generic architecture to represent knowledge about plans and contingencies that occur during planning. The Redux concepts build on artificial intelligence planning approaches, but can easily be applied to process planning.

One of the main services the Redux model provides is dependency-directed backtracking. Both CoMo-Kit and Procura (see below) use Redux for dependency management.

CoMo-Kit defines and implements a methodology for project planning, and incorporates an interpreter for plan enactment. Its architecture consists of two parts: a Modeler for defining and implementing the ontology and a Scheduler for project plan execution. By using the Redux model, CoMo-Kit provides dependency tracking for technical roles.

Procura [9] is a Project Management model which allows planning and scheduling of agent-based design projects in a hierarchical top-down approach. Its main difference to CoMo-Kit is the issue of scheduling: Procura integrates Critical Path techniques in the dependency tracking mechanism. Also, Procura is an agent-based approach in contrast to CoMo-Kit's client-server architecture.

As a result of the research project „SFB501: Development of large systems with generic methods“[1], the MILOS approach [5] was conceived. MILOS is a process-centered modeling language strictly focussing on software development processes. MILOS integrates the benefits of CoMo-Kit and MVP-L (Multi-View Process Modeling Language). MVP-L [4] was designed to improve the processes by means of descriptive modeling, analyzing these models and saving the models in an experience factory. It provides role specific views on the project plan and guides the different roles in performing their tasks.

We cooperate with the Department of Computer Science University of Calgary, where an object-oriented database is being integrated in the existing CoMo-Kit architecture, and the project management tool AutoPLAN is being included as a legacy tool [6].

The integration of legacy software is a point we are also discussing. We have implemented an export interface for CoMo-Kit, which allows us to export project information from CoMo-Kit into other tools via ODBC. This allows legacy tools to access the project information stored in CoMo-Kit.

Another point we are currently working on is the integration of the measurement planning tool GQM-DIVA [12], a tool for defining, interpreting and validating measurement plans.

6 Related Research

Process sensitive Software Engineering Environments (SEEs) have been a research topic during the last years. Several approaches have been published which support the

1. see http://wwwsfb501.informatik.uni-kl.de:8080/

technical software process, e.g. SPADE, MARVEL, MERLIN, etc. See [8] for an overview of these earlier approaches, and [5] for a comparison between our research and these ideas.

More recent process sensitive environments are being introduced as part of the Arcadia[1] project; for example, [3] describes an approach focussing on the coordination of technical roles in the software development process.

The OzWeb[2] project is another example for current research on project coordination in Software Engineering. [10] describes a WWW based SEE with tool support for technical roles.

An example for a web-based approach on coordination is the METEOR[3] project at University of Georgia [13]. This work distributes a workflow management system over the internet, using CORBA as a means of communication as well as for transaction management.

Our research builds on the measurement based approach on software process support as described in [11]

Literature

[1] V. R. Basili, G. Caldiera, and H. D. Rombach: *Measurement*, in Encyclopedia of Software Engineering (John J. Marciniak, ed.), vol. 1, pp. 646--661, John Wiley Sons, 1994.

[2] V. R. Basili, G. Caldiera, and H. D. Rombach: *Experience Factory*, in Encyclopedia of Software Engineering (John J. Marciniak, ed.), vol. 1, pp. 469--476,John Wiley Sons, 1994.John Wiley Sons, 1994..

[3] G. A. Bolcer and R. N. Taylor: *Endeavors: A Process System Integration Infrastructure* in Proceedings of the Fourth International Confernce on the Software Process, Brighton, England, December 1996.

[4] A. Bröckers, Ch. Differding, and G. Threin: *The role of software process modeling in planning industrial measurement programs*. In Proceedings of the Third International Software Metrics Symposium, Berlin, March 1996. IEEE Computer Society Press.

[5] B. Dellen, F. Maurer, J. Münch, M. Verlage: *Enriching Software Process Support by Knowledge-based Techniques,* International Journal of Software Engineering and Knowledge Engineering, 1997.

[6] B. Dellen, F. Maurer: *A Concept for an Internet-based Process-Oriented Knowledge Management Environment*, to appear int he Proceedings of the KAW'98, Banff, Canada , 1998.

[7] B. Dellen, F. Maurer: Change impact analysis support for software development processes, To appear in the Journal of Applied Software Technology, International Academic Publishing, 1998

[8] .P. Garg and M. Yazayeri: *Process-Centered Software Engineering Environments: A Grand Tour*, in Software Process, (Fugetta and Wolf, ed.), 1996.

[9] S. Goldmann: *Procura: A Project Management Model of Concurrent Planning and Design*, Proceedings of WET-ICE '96, IEEE press, Stanford, USA, 1996.

[10] G. E. Kaiser, St. E. Dossick, W. Jiang, J. Jingshuang Yang and S. X. Ye,: *WWW-based Collaboration Environments with Distributed Tool Services*, to appear in World Wide Web, Baltzer Science Publishers.

[11] C. M. Lott, B. Hoisl, and H. D. Rombach: *The use of roles and measurement to enact project plans in MVP-S*, in Proceedings of the Fourth European Workshop on Software Process Technology (W. Schäfer, ed.), (Noordwijkerhout, The Netherlands), pp. 30--48, Lecture Notes in Computer Science Nr. 913, Springer--Verlag, Apr. 1995.

[12] M. v. Maris: *GQM-DIVA, a tool for defining interpreting and validating GQM plans*. Master's thesis. Department of computer science, University of Kaiserslautern, 67653 Kaiserslautern, Germany, 1995.

[13] J. Miller, A. Sheth, K. Kochut, and D. Palaniswami: *The Future of Web-Based Workflows*, Proc. of the International Workshop on Research Directions in Process Technology, Nancy, France, July 1997.

[14] Ch. Petrie: *Planning and Replanning with Reason Maintenance*. PhDthesis, University of Texas at Austin, CS Dept., 1991. (MCCTR EID-385-91).

[15] I. Sommerville: *Software Engineering*, 3. Auflage, Adison Wesley, 1989.

1. see http://www.ics.uci.edu/~arcadia/
2. see http://www.psl.cs.columbia.edu/edcs/brochure.html
3. see http://lsdis.cs.uga.edu/workflow/