MILOS: A Model of Interleaved Planning, Scheduling, and Enactment

Sigrid Goldmann, Jürgen Münch, Harald Holz

Universität Kaiserslautern, Postfach 3049, 67653 Kaiserslautern, Germany
{sigig, muench, holz}@informatik.uni-kl.de

1 Introduction

Software development processes are highly creative, and subject to frequent changes - making it hard to plan and schedule these processes in advance. The recent trend towards distributed software development and virtual software corporations adds another aspect to this problem: Distributed projects cannot be planned centrally, especially if the involved parties are individual companies with their own areas of responsibility. On the other hand, certain aspects of the overall plan and schedule (e.g. will the project milestones be met? when will Company X finish a product needed as an input for Company Y's tasks?) need to be available to all concerned parties.

In this paper, we present an approach to support distributed planning and scheduling, as well as the subsequent (also distributed) plan execution, in one system. The system will support the distributed planners and schedulers by providing task agendas for them, stating who needs to plan which tasks, and sending change notifications and warnings, if a planning or scheduling decision needs to be updated. The plan built using these mechanisms is then enacted by a workflow engine in the same system. This approach enables us to support interleaved planning and plan enactment, allowing the user to change the plan and schedule while the project is already under way. Deviations of the actual project enactment from the plan and schedule can automatically be detected, and necessary notifications will be sent to the concerned planner(s). This again facilitates the task of keeping the plan up to date, avoiding the complete invalidation of the plan as is often the case in conventional projects soon after enactment has started.

Section 2 introduces the MILOS model of interleaved planning, scheduling, and enactment support, and summarizes the ontology of terms used in this model. Section 3 describes an algorithm of how planning, scheduling, and plan enactment can be interleaved, and gives an example scenario to clarify the problem and our proposed solution. Section 4 gives a short overview over the techniques MILOS uses to support distributed project enactment, and shows how these same techniques can be used to facilitate planning and scheduling, as well as interleaved planning and enactment. Section 5 gives an overview over the current state of implementation, section 6 describes related research, and section 7 summarizes the paper and names the areas in which further research is necessary in order to make the approach feasible for virtual corporations.

2 The MILOS Ontology

In this section, we briefly describe our use of the terms modeling, planning, scheduling and enactment in this paper, and introduce our concept of meta tasks.

2.1 The Process Model and Project Plan

A MILOS *process model* contains general information about how software processes are done. For each kind of software development process (requirements analysis, system design, etc.) that can occur in a certain kind of development project (e.g. accounting software development, regulation software development), a general description is given, and its input and output *products*, pre- and post*conditions*, and general *attributes* (e.g. estimated process duration, effort spent on the process) are defined.
For each process, different *methods* can be specified, describing alternative refinements:

- An *atomic method* defines a way to solve the process directly. For example, code inspection can be done in an

ad-hoc way, or using a formal reading technique.

- *Complex methods* refine a process into a number of sub-processes, e.g. a component development process is composed of a component design, a design inspection and a component implementation process.
  In complex methods, the data flow between the subprocesses can be specified by mapping the subprocesses' inputs to outputs of other subprocesses in the same method. The subprocesses introduced by a complex method can further be refined by their own complex methods, resulting in an and/or tree with a process as root-node.

For each method, a set of skill requirements is defined, which describe the qualifications necessary to use that method to work on a process. For example, if the method "use Java" is selected for a component implementation process, the programmer assigned to that process should have general knowledge about object-oriented programming, as well as skills in the programming language Java.

Resource models describe types of project resources or agents (i.e. people or tools), with the resources' skills and roles. These models can be used during process modeling to describe the type of resource that is needed to work on a process or apply a method.

The general process information described above is stored in a company-specific "experience base", to be reused in specific projects. Other information, like quality models[1] that have proved accurate in past projects, are also stored in this experience base, and can be used to plan a project.

A MILOS *project plan* or *schedule*[2] is built by

- selecting appropriate processes from the experience base, and inserting them into the plan,
- mapping certain processes' inputs and outputs onto each other (i.e. defining a product flow)
- selecting applicable development methods according to the characteristics of the organizations (e.g. familiarity with specific methods) and the goals of the project (e.g. budget limitations),
- instantiating variables in pre- and postconditions (e.g. using a quality model about the relative efforts of different development activities[3] to calculate the absolute effort as desired values for the project activities, which should not be exceeded. Theses values are assigned to the corresponding processes' "effort" attributes),
- allocating resources to the processes according to the resource properties specified in the respective process models, and
- time scheduling with these resource allocations in mind.

The first four of the above activities (selection of processes and methods, defining the data flow and control flow) we summarize under the term *planning*. The latter two activities (time and resource assignment) we call *scheduling*.

The resulting plan contains instantiated processes, their subprocesses, and the data flow and control flow between them. It also contains scheduling information like the processes' assigned agents and scheduled start and end times. Certain *dependencies* (e.g. between a process and its subprocesses, between a process and its predecessors) can be extracted from the plan, and be used to track the necessity of change notifications during plan enactment. A more detailed description can be found in [5].

2.2 The Meta Plan

While a project plan contains those *tasks* that lead to the creation or modification of (software) products, i.e. the processes, the *meta plan* contains tasks who's goal is the construction or modification of the project plan. Those tasks that modify the plan we call *meta tasks*, while we refer to processes also as *development tasks*.

The two meta tasks that we will deal with in this paper are *planning tasks* and *scheduling tasks*. Planning tasks occur on user agendas when a process needs to be refined, its data flow needs to be specified (i.e. its input and output parameters need to be matched to the inputs and outputs of other processes in the plan), or its control flow needs to be refined (i.e. values need to be assigned to open variables in a process' precondition or postcondition). A scheduling task is generated when resources are to be assigned to a process, and its scheduled start and end times have to be defined.

Similar to the project plan, the meta plan can be modeled as a hierarchy of tasks, in correspondence to the hierarchy of processes in the plan. Like the plan, the meta plan can be used to automatically identify dependencies between the different tasks in the meta plan (e.g., when a task is removed from the plan, its subtasks do not need to be done any more). Additionally, there are dependencies between the meta plan and the plan: Whenever a new process is inserted in the project plan, it needs to be planned (i.e. further planning activities are required; see above) and scheduled, therefore necessitating the insertion of certain planning and scheduling tasks into the meta plan. On

1. Quality models map measurable influence factors to quality factors of interest.
2. In this paper, we use the terms *plan* and *schedule* synonymously
3. For example, the quality model might state that 40% of the total effort should go into requirements engineering, 20% into system design, etc.

the other hand, planning a process can result in the insertion of (sub-)processes in the plan. By updating the plan and meta plan according to the rules we identified, and sending notifications whenever replanning becomes necessary, the MILOS system will be able to facilitate keeping the plan and schedule up to date, and checking the project progress against defined milestones and deadlines.

3 An Algorithm for Interleaved Planning, Scheduling, and Enactment

In this section we outline an algorithm for the simultaneous enactment of plan and meta plan, i. e. for interleaved planning, scheduling, and plan enactment. A short example scenario is given which demonstrates the interaction between plan and meta plan. The following algorithm abstracts from details and sketches the principle procedures. Especially the collaboration among planners at different development sites is not included in the description, although necessary if project plan changes concern external product and control flow.

1. Perform meta tasks before project start
   Goal: create initial project plan
   Role: planner
   We assume that at the start of a project a initial project plan is created which is tailored to the specific needs and characteristics of the development organization. Existing experience (e. g. process fragments, quality models) should be used and adapted if applicable and valid in the context of the project. The initial project plan may be instrumented with measures. Furthermore, in the case of distributed planning, the interfaces among different development sites (especially concerning product and control flow) have to be defined.
   The initial plan is usually not described in all details and may be refined, modified or extended during project enactment.

2. Perform meta tasks during plan enactment
   Goal: plan and schedule the next level of the project plan
   Roles: planner, modeler
   - 2.1 Plan next level
     - 2.1.1 Select appropriate methods for the next level due to the exit criteria (i. e. the goal specification) of the process to be performed.
     - 2.1.2 If no methods exist which are suited to fulfil the process specification the missing method can be modeled
     - 2.1.3 Adjust variables in the pre- and postconditions according to the process specification
     - 2.1.4 Inform planner of the parent process if
       a) it is not possible to select or model an appropriate method
       b) the process specification must be adjusted
   - 2.2 Schedule next level
     - 2.2.1 Assign personnel to the selected methods
       a) assign planners to subprocesses (in case of complex methods)
       b) assign developers to atomic methods
     - 2.2.2 Determine start- and end dates according to the process specification and negotiate them with the planners/developers

3. Perform development tasks during project enactment
   Goal: create/modify output products
   Role: developer (e. g., requirements engineer, designer)
   - 3.1 Start method execution.
     If the method is in state `performed` and the exit criteria is met)
       then assign value to output product
     If ((method is in state `performed`) and (exit criteria is not met)) or (invariant does not hold)
       then inform the responsible planner for this method
       [an invariant is a condition which should be true during the method enactment]

A short scenario is given to illustrate this algorithm (see Fig. 1). Let us assume that Alissa is the responsible planner for all subprocesses of the complex method “component development”. Her task is to plan this subprocesses according to the exit criteria (e. g., effort should be less that 50 hours). She selects appropriate methods for the processes “create component”, “component validation” and “integrate component”. These methods are selected in a

way that it is possible to fulfil the exit criteria of the process “component development”. As an example, Alissa selects the complex method “white box test”, instantiates the variables in the criteria for this method and assigns Susan as the responsible planner for the processes “generate test cases” and “perform tests”. Additionally, Susan receives planned start- and end times for this process. It is Susan´s task for further planning. She assigns due to the qualifications of the available personnel Margret and Paul as developers to the subprocesses and negotiates with them the planned start- and end times. During the method execution Margret recognizes that it is not possible for her to complete her process before the planned end date. The reason might be that the complexity of the component to be tested is too high. Consequently, she informs Susan. Susan´s task is now to change the start- and end times and to inform Paul that he will receive the input product later. If this rescheduling is not possible, Susan can model new methods for the processes “generate test cases” and/or “perform tests” and assign them to appropriate developers. If Susan doesn’t have a possibility to reach the exit criteria of the process “component validation” she informs Alissa. Alissa might then select another method (e. g. statistic test) for execution and assign a new planner to this method.

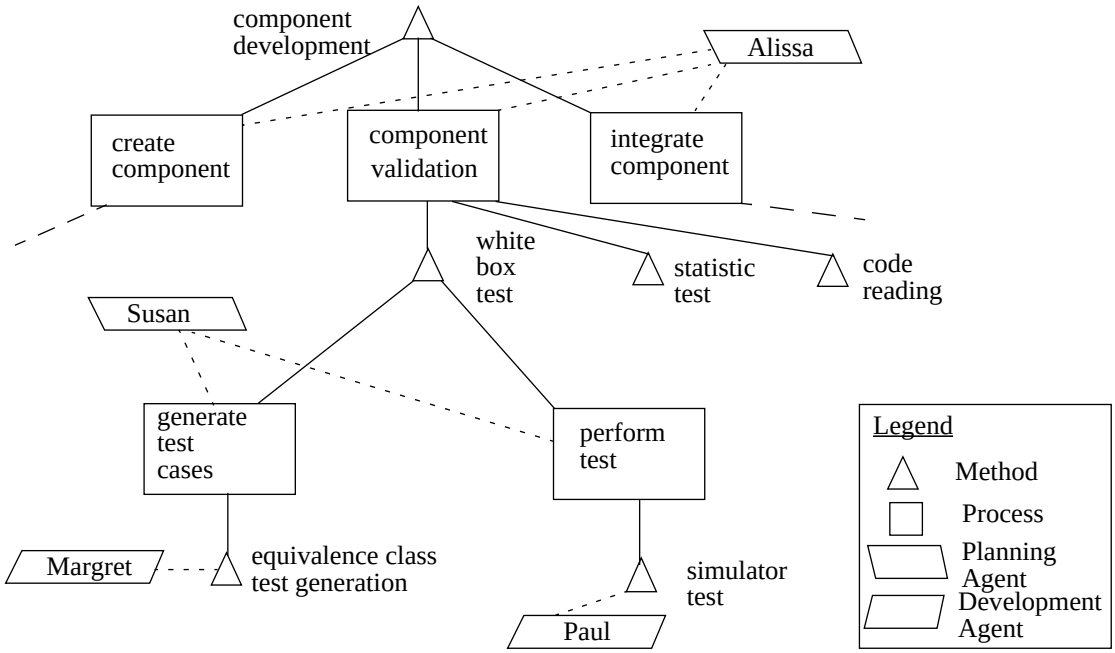

Fig. 1. Example plan (product flow omitted)

4 Dependency Management in the MILOS Project Plan and Meta Plan

In this section, we describe the techniques the existing MILOS system uses to facilitate the enactment of an existing plan, and show how these same techniques can be used to facilitate planning and scheduling, i.e. the enactment of the meta plan.

4.1 Enactment of the Project plan

Execution support in the existing MILOS system has two important tasks:

- Provide to-do lists for agents, and generate notifications to the concerned agents whenever a process can be started (or needs to be restarted), or when something of interest (e.g. an input product, a scheduled time, or a process definition) has changed.
- Provide feedback to the planner about plan violations during execution, and allow the model and plan to be changed during process execution, triggering change notifications to the concerned agents.

The first point provides guidance to the agents, and thereby helps ensure that the “real world” process conforms with the plan. The second, on the other hand, allows the plan to be adjusted when necessary, and therefore prevents the plan to become obsolete when execution does not follow the plan in spite of the guidance our system provides.

Concerning agent notifications, our system supports different kinds of notification techniques. On the one hand, we provide standard notifications like escalation mechanisms (e.g. user notification on approaching deadlines). On the other hand, we generate notification dependencies from the project plan, and allow project participants to express interest in specific information. To implement these notification dependencies, we use Event-Condition-Ac-

tion (ECA) rules that can be based on product and process-specific events. For example, if the precondition of the component testing process demands that the component requirements document should be complete, our system would automatically generate the following rule:

Event: document *component requirements* completed
Condition: the *component test* process has been assigned to *Agent x*
Action: Notify *Agent x*

(See [5] for more details concerning our use of ECA rules.)

Feedback to planners is provided by keeping a project trace, and checking the trace data (like actual start and end times) against the plan. If a process' execution data deviates from its planned values, the assigned agent as well as the responsible planner are notified, so that necessary replanning can be performed. These notifications are also triggered by corresponding ECA rules.

4.2 Enactment of the Meta Plan

Not only do modeling and planning depend on enactment data, but there are also dependencies between different modeling and planning activities that concern the same process or subplan. Once these activities are performed in a decentralized way, support is needed to coordinate modeling and planning activities as well as execution activities. For example, the removal of an output parameter in a process model might necessitate replanning in case the output is needed somewhere else in the plan. If these different modeling and planning activities have been distributed between different people, notifications have to be triggered in order to inform all concerned people of the necessity of replanning or re-modeling their processes.

Other examples for model or plan changes that necessitate notifications are:

- A new process is inserted in the plan, which needs an input that no other process in the plan produces. All planners responsible for parts of the plan that might be modified to produce the product need to be notified. The same holds when an additional input is added to a process already in the plan.
- An input is deleted from a process in the plan, making it possible to start working on the process at an earlier time than expected (and scheduled). The concerned planner should be notified of the arising opportunity.
- Conflicting methods are selected for different processes, e.g. the method *Implement in Cobol* is selected for a component which is to be designed using the method *Object-oriented design*.
- A resource with a rare skill becomes available, opening an opportunity to reschedule.

As mentioned in [12], modeling and planning can be seen as a different kind of (meta) process, and can be handled as such. If modeling and planning (and re-modeling and replanning) are seen as tasks in the meta process of "managing a software project", these *meta tasks* can be assigned to agents (i.e. modelers and planners), just like development processes are assigned to resources with the appropriate skills. The meta process can be modeled, and ECA rules can be extracted from that meta process. For example, when the output of the system design process is changed from an OMT document to a UML document, the following ECA rule will be triggered (among other rules):

Event: output type for process *system design* has changed
Condition: process *component design* needs *OMT design document* as an input
AND planning task for process *component design* has been assigned to *Agent y*
Action: Notify *Agent y*

In other words, the same mechanisms can be used to coordinate modeling and planning that we already apply to execution: from the notification mechanisms and ECA rules to agendas for meta tasks, all coordination concepts developed for enacting the software process itself can also be utilized in order to facilitate the meta process of modeling and planning the software process.

5 State of Implementation

Currently, the MILOS environment consists of several components: a resource pool, a process modeling component, a project plan management component, and a workflow engine. These components are responsible for maintaining the information concerning a company's resources, process models, a project's plan as well as its current state and products, respectively, and are linked via a change management mechanism. A detailed description of the architecture is given in [9].

The workflow engine enacts the current plan stored in the project plan management component. The individual planning activities that were necessary to construct the plan are of no concern to the engine; it is solely responsible for the coordination of development tasks. Thus, the current implementation of MILOS does not yet provide support to explicitly model the meta process of modeling and planning the project, and therefore the different meta

tasks cannot be assigned to the appropriate planners. With respect to the above scenario, this means that while the development agents are notified individually of the changes that concern them, MILOS cannot yet identify which of the two planners needs to be notified of the replanning necessity. The current system version handles this problem by notifying *all* involved planners whenever the necessity of replanning occurs.

However, the chosen architecture allows us add a "plan engine" as a new component. As the workflow engine provides to-do lists of development tasks to each development agent, this plan engine will provide to-do lists of meta tasks to planners. Performing one of these meta tasks will result in a change of the project plan. Besides supporting immediate execution of a task, the user interface for planning agents will also allow delegation of meta tasks to other (planning) agents, thereby facilitating distributed planning.

The "meta plan" to be enacted by the plan engine will be induced from the current project plan via the MILOS model of planning, e.g. for each development task *t* there will be a corresponding task `schedule(t)`, for each unmapped process input *i* a planning task `create_mapping_for(i)` etc.

Likewise, default dependencies between meta tasks can be induced from the project plan: a change in a task's scheduling will result in the creation of `reschedule(t)` meta tasks for certain successor tasks *t* (depending on the particular scheduling change). As for development tasks, these dependencies will be managed in the form of ECA rules.

The MILOS system so far has been implemented in Java, using the object-oriented database GemStone/J 2.0 as an Enterprise Java Bean (EJB) server that provides transaction management and persistency services. This server manages the process model and project plan, and provides support for project enactment. Clients are responsible for modeling, planning and executing software development processes. They are stand-alone applications or Java applets which access the server via HTTP or using a Java Remote Method Invocation (Java RMI) interface. We provide tool support for applet clients by using the built-in capabilities of Web-Browsers to download files and start the correct tool at the client's host. In other words, products that have to be edited in order to complete a process are specified by a URL. The Web-Browser executing the client applet accesses this URL, downloads the file, and starts the associated tool to edit the file. The edited file can then be saved on the client computer, or uploaded back to the server.[1]

The data exchange with MS-Project has been implemented using the MS-Project API, which allows import and export of MS-Project plans in a predefined exchange format. Our system is able to import and export plans in the same format, and to identify similarities and differences between imported plans. This makes it possible for us to model a project in MILOS, import it into MS-Project, change it in MS-Project, and import it back into MILOS, and vice versa.

6 Related Research

Our work bears similarities to several areas of research, particularly project management tools, workflow management approaches, and process modeling and enactment research. Commercially available project management tools like MS-Project and Autoplan support project planning and scheduling, but provide little or no enactment support. A project management system that does provide both planning and execution support is the Mesa/Vista Enterprise tool, an environment for collaborative project execution and management. It provides distributed access to project data, as well as version and configuration management, but it does not include any change notification services.

Workflow management tools like Staffware, FlowMark, or TeamWARE concentrate on project execution and provide little or no support for process modeling and project planning. In particular, plan changes during enactment require a complete restart of the project in most workflow management tools.

The approaches most similar to our work can be found in the area of process modeling and enactment research. Most approaches in that area provide (web-based) modeling and enactment functionality, as well as some support for dynamic plan changes and change notifications. However, most of these approaches do not provide project planning and management support, like resource allocation and time scheduling for tasks in the project. Below, we briefly describe a number of approaches in the area of process modeling and enactment research.

Endeavors [2] is a support system for distributed execution of (workflow) processes. Endeavors provides support for dynamic process changes, and is currently being extended to support WWW protocols.

Serendipity [7] is a process modeling and enactment environment that supports collaborative modeling as well as execution of software processes. Change notifications are sent, using an event-handling concept similar to our approach. Several external tools have been integrated in the Serendipity system.

1. We use signed applets as specified in the Java 1.2 security model.

OzWeb [8] is a web-based system that supports multiple users who are grouped together into collaborative teams. OzWeb provides a framework that supports the storage of retrieval of information in a "referential hyperbase", and provides some notifications based on dependencies extracted from a process model.

EPOS [11] is a Software Engineering Environment with emphasis on process modeling, software configuration management and cooperative work support. The EPOS system is based on an underlying database, which provides versioning functionality and transaction management, controlled by an application-specific process model.

The SPADE [1] project aims at defining and developing a software engineering environment for software process modeling and enactment. Its process modeling language is based on a high-level Petri net formalism. The SPADE research also includes techniques to deal with process evolution during enactment.

Like the MILOS model, the Procura [12] approach assumes that the same mechanisms that are used to coordinate the enactment of design projects can be used to support planning and scheduling. The Procura approach builds on the Redux model of flexible design [13], which can be used to track dependencies between design decisions, and send change notifications to the concerned agents when necessary.

Multi-view approaches in the area of process modeling allow the distributed modeling of objects in different styles and representations (e.g. control-flow view, abstraction hierarchy view, role-oriented view). These approaches can be classified according to their integration mechanisms. One class is characterized by separate modeling of different views and subsequent integration. A representative of this class is the MVM approach (multi-view modeling) [16]. This approach is based on role-specific views, which are modelled independently using the formal process modeling language MVP-L [3]. Finally, the integration of views is performed with similarity and consistency analyses and the creation of a comprehensive software process model.

The other class of approaches is characterized by the distributed modeling of a common model. This implies the permanent application of consistency checks and updating operations. A typical approach of this class is the MUVIE approach [14]. Here, each view defines a focus on an underlying graph structure model. The modeller only handles those parts that pertain to a specific view. The underlying semantics which guide incremental changes are expressed by a graph model and graph replacements.

7 Conclusions and Future Work

MILOS is a process modeling and enactment system which not only supports modeling and enactment in the same system, but also provides project management functionality in the form of planning and scheduling support. This allows us to guide project execution according to the project plan and process model, as well as to keep the plan up-to-date by feeding back enactment information into the plan. MILOS' flexible workflow engine allows the model and plan to be changed during project enactment, and provides support for process restarts whenever necessary. It provides task agendas for the agents enacting the plan, and notifies the agents when a process assigned to him/her becomes executable. Moreover, the MILOS model uses ECA rules to model dependencies between the processes in the plan. Whenever changes occur, these rules are evaluated, and change notifications are sent to those agents who are concerned by the change. These same mechanisms can be used to coordinate planning and scheduling as well as plan execution, and allow to track dependencies between the meta tasks of planning and scheduling on the one hand, and the enactment of the plan on the other hand. In this paper, we presented a flexible model of how these dependencies can be specified and tracked. A default (meta) model describes standard dependencies between planning, scheduling, and plan enactment. This model can be modified and extended to state project specific dependencies, for example the fact that a certain planning decision depends on a product produced during project enactment.

In the future, we will investigate the possibilities of extending our meta model by additional meta tasks. For example, quality assurance, and the related task of planning and executing appropriate measurements in order to ascertain the quality of process and product, are tasks that will be included in a future version of the MILOS model and system.

In order to satisfactorily supporting software development in virtual corporations, the MILOS system will have to be extended by security concepts. Also, decentralization of data storage, and the distribution of our workflow and planning engines will be central topics in our future research.

Acknowledgments

The current MILOS system was developed and implemented in cooperation with Prof. Dr. Frank Maurer' research group at University of Calgary. Barbara Dellen, Boris Kötting, and Fawsy Bendeck were involved in the conceptual work as well as the implementation of the MILOS system. The work was supported by NSERC, Nortel, the University of Calgary, and the DFG with several research grants.

Literature

1. 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).
2. G.A. Bolcer and R. N. Taylor: Endeavors: *A Process System Integration Infrastructure*. in Proceedings of the Fourth International Conference on the Software Process, Brighton, England, December 1996.
3. 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.
4. Bill Curtis and Marc I. Kellner and Jim Over: *Process Modeling*. Communications of the ACM, Vol. 35, No. 9, September 1992.
5. B. Dellen, F. Maurer: *Change impact analysis support for software development processes.* Journal of Applied Software Technology, International Academic Publishing, 1998.
6. B. Dellen, F. Maurer, J. Münch, and M. Verlage: *Enriching Software Process Support by Knowledge-based Techniques*. In Int. Journal of Software Engineering and Knowledge Engineering, Volume 7, No. 2, pp. 185-215, 1997.
7. J.C. Grundy and J.G. Hosking,: *Serendipity: integrated environment support for process modeling, enactment and work coordination*. Automated Software Engineering: Special Issue on Process Technology 5(1), January 1998, Kluwer Academic Publishers, pp. 27-60.
8. G.E. Kaiser, St.E. Dossick, W. Jiang, J. Jingshuang Yang and S.X. Ye,: *WWW-based Collaboration Environments with Distributed Tool Services*. World Wide Web, Baltzer Science Publishers (to appear).
9. F. Maurer, B. Dellen: *A Concept for an Internet-based Process-Oriented Knowledge Management Environment.* Proceedings of the KAW'98, Banff, Canada, 1998
10. F. Maurer, G. Succi, H. Holz, B. Kötting, S. Goldmann, B. Dellen: *Software Process Support over the Internet.* submitted to ICSE99, Formal Demonstration Track.
11. M.N. Nguyen, A.I. Wang, R. Conradi: *Total Software Process Model Evolution In EPOS*. Submitted paper for 4th ICSP, 1996, Brigthon, UK.
12. Ch. Petrie, S. Goldmann, A. Raquet: *Agent-Based Project Management*. to appear in Springer LNAI 1500 special volume on Artificial Intelligence Today.
13. Ch. Petrie, Th. Webster, M. Cutkosky: *Using Pareto Optimality to Coordinate Distributed Agents*. AIEDAM special issue on conflict management, Vol. 9, pp 269-281, 1995.
14. Peter Rösch, "User Interaction In A Multi-view Design Environment". IEEE Symposium On Visual Languages (Vl'96), Proceedings, Pp. 316-323, Boulder, Colorado, Sept. 3-6, 1996. Usa, Ieee Computer Society Press, ISBN 0-8186-7508-x.
15. 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.
16. Martin Verlage, "An Approach For Capturing Large Software Development Processes By Integration Of Views Modeled Independently". 10th International Conference On Software Engineering And Knowledge Engineering (Seke98), June 1998.