

A Three Level Framework for Process Support: The MOWAHS Approach

Alf Inge Wang
alfw@idi.ntnu.no

Carl-Fredrik Sørensen
carlfrs@idi.ntnu.no

Reidar Conradi
conradi@idi.ntnu.no

Dept. of Computer and Information Science,
Norwegian University of Science and Technology,
Sem Sælands vei 7-9, NO-7491 Trondheim, Norway,
Phone: +47 73594485, Fax: +47 73594466.

Abstract

A common assumption for many process-centred support environments is that they provide the same process support at different levels of the organization. We believe that the required process support from one level to another of an organization varies. In this paper we propose a framework that divides process support into three levels: Individual, group and team level. Further, we characterize each level and describe the required process support and type of process modelling language for each level. We also look at what interfaces are needed between the levels. Finally we use the framework to characterize an eXtreme Programming (XP) development process. Our framework focuses only on the project level of software development and does not consider management processes above this level.

Keywords: *Software process support, software process modelling, process-centred support environment, cooperative processes, and eXtreme Programming.*

1 Introduction

From the start of the software engineering era software development processes have been modelled to understand the process, to guide people involved in the process, to partially automate the process, and to improve the process. Many process models and process-centred support environments (PSEs) have been made with mostly the same assumption, that the same process support should be provided for every level in an organization [3, 6, 7, 9]. If we look at software development processes at different organizational levels, we discover that the characteristics of the required process support vary dependent on the level.

The software process support required at the *individual level* is often process guidance (especially for inexperienced personnel), activity reminders and progress in calendars, and automation of repeatable tasks.

For *groups* being at the next organizational level, the process support can be characterized as Cugola and Ghezzi states in [4]: "Human-centred processes are characterised by two crucial aspects that were largely ignored by most software process research: They must support cooperation among people, and they must be highly flexible". Software development processes are human processes and at a group level the main focus will be on cooperation and coordination between the participating roles.

For *teams*, the top most organizational level, the required process support will focus on managerial issues and deal with heterogeneous tools and models. A team is typically an organizational unit responsible for specific parts of a software lifecycle like design or testing. It can be a sub-organization (department) or a virtual organization specific for a project.

In this paper we will describe a three level framework where we have identified the process support at each level and also discuss the type of process models required at each level. This work is a result of a project called MOBILE Work Across Heterogeneous Systems (MOWAHS) [11] where the main focus is on modelling and supporting mobile work processes.

The rest of the paper is organized as the following: Section 2 describes the framework, Section 3 applies the framework to a part of a software process, Section 4 relates our work to similar research, and Section 5 concludes the paper.

2 A Three Level Framework for Process Support

This section presents a framework where the required process support are grouped and identified in three levels: Individual, Group and Team Process. Figure 1 shows an overview of the process model elements; the required external resources used by the PSE and the required process support at individual, group and team process level. In this

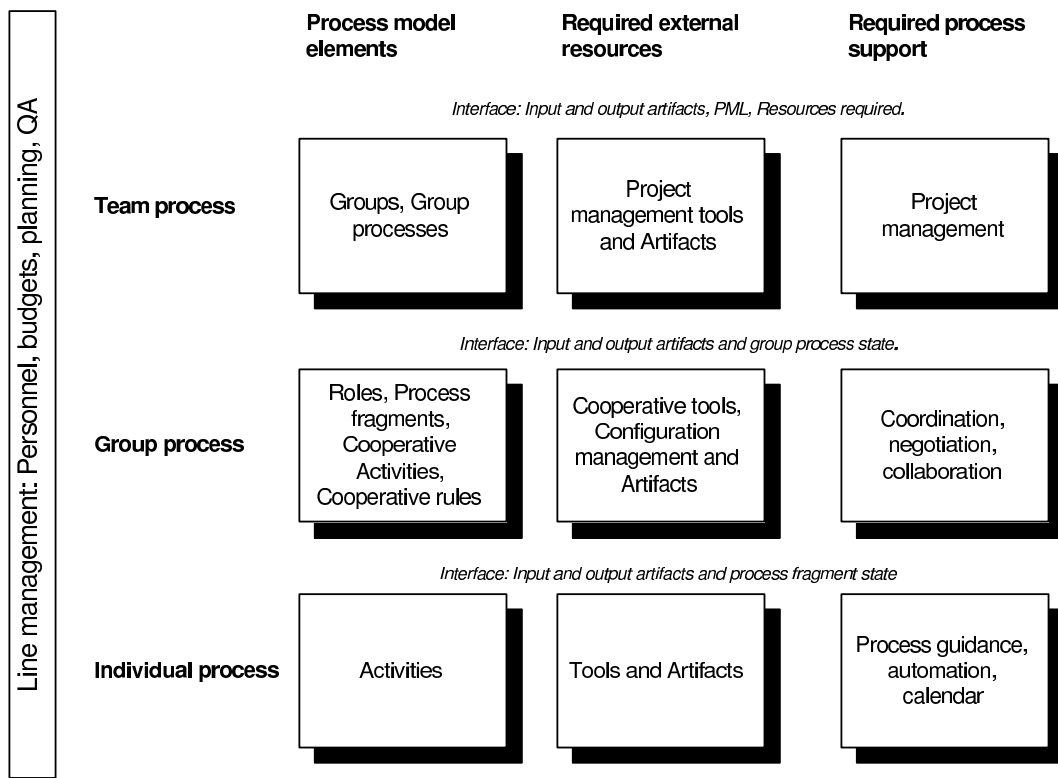


Figure 1. An overview of the MOWAHS three level framework for process support.

framework we have captured the main differences between the various levels of a software development process. Each process level will be described in more details in the following sections. In addition to typical project processes, line management processes are important for e.g., management of available resources, project organization, and quality assurance (QA). The line management affects the processes at all three levels of the project organization. Processes related to line management is not the focus of this paper since they are normally not limited in time. Our focus is the project processes.

Our framework does not explicitly define workspaces at the three levels. We assume that workspaces are available for all levels where the personnel can access tools and artifacts. A repository providing configuration management can also manage the artifacts.

2.1 Individual Process

The lowest level of the framework is the *individual process level*. The main focus at this level is the software developer and his/her tasks. A process for an individual actor of a software process typically consists of a set of activities related to the roles that the actor plays. It is very important that the process support is tailorable and configurable,

making it possible for the actor to fit the work to personal preferences. The required process support is very dependent on the experiences of the individual actor. An inexperienced person would typically need process guidance that will help this person to know what to do next, what procedures to follow, what tools to use, where to find documentation etc. However, for an experienced person that knows all the basics, extensive process guidance would be more a hindrance to do the work effectively. Independent of the experience level of the actors, the process support should provide quick and easy access to the required tools and artifacts to perform the activities. The process should be integrated with personal electronic calendars to enable activity states and deadlines to be accessible on personal computers, personal data assistants (PDAs), mobile phones and even digital watches. This means that the execution environment for the process support also must be capable of communicating and utilizing mobile devices and networks.

Our framework identifies *activities* as the main building block for individual process models. There are several reasons for this. Firstly, activities or tasks are commonly used in calendar software for personal computers, PDAs, mobile phones and other electronic devices. This makes it easier to integrate the process support with all devices that a person uses. Secondly, project-planning tools usually produce

some kind of activity networks where activities can be delegated to individual actors in a project. Thirdly, it is most common to think of individual processes in terms of activities. Since it should be easy to tailor the individual process, it is important that the process and the process model are easy to comprehend by the actor actually using the process. In [14], an experiment indicates that activity-based modelling is easier than role-based modelling. This is especially true for inexperienced process modellers.

We define in our framework a collection of related activities to be a *process fragment* (PF). The process fragment is the container element used in our framework to specify a set of related activities (see Figure 2). Further, a process fragment is one or more activities that are related by pre-order relationships. An actor will typically at the individual process level have several process fragments from the same project or possibly process fragments from other projects. A process fragment is the main process model building block for the group process described in the forthcoming section.

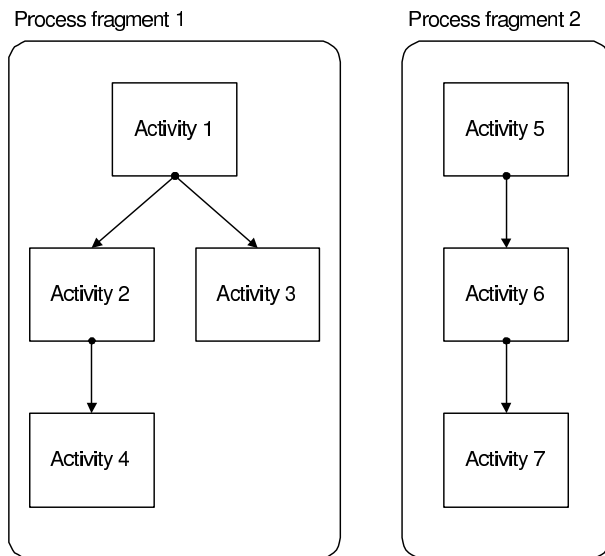


Figure 2. A simple process model consisting of related activities grouped as process fragments.

The individual process given to a software developer is usually a template of a process where all the necessary activities are defined. The software developer uses the template to fill in missing information into the model, and tailor the model to his/her purpose and preferences. During the process enactment, the software developer might also add or delete activities depending on how the project proceeds (e.g., change of project plan, process environment, or project goal).

2.2 Group Process

The middle level of the framework is the *group process level*. The main focus at this level is the cooperative aspects of the software process. The cooperative process involves coordination of activities from several actors or roles, cooperative activities where two or more persons participates at once, and coordination and negotiation of artifacts and resources. Whenever several persons share artifacts, configuration management (CM) is an important part of the support and coordination of the process. The CM environment must be integrated with the PSE to provide negotiation about conflicting resources and smoothness handover of output artifacts from the process. Further, the group process defines the synchronisation points for the individual processes from several roles or actors. In addition the group process involves cooperative activities like distributed brainstorming, electronic voting, collaborative authoring, and conflict management. Cooperative activities have very different characteristics compared to individual activities. While for the individual activities the main emphasis is on tasks to be done, cooperative activities are all about interaction between roles. This means that the process support for cooperative activities must provide an infrastructure to enable the involved roles to interact in an effective way and to enable flexible exchange of artifacts. We have proposed the use of a cooperative agent framework to provide cooperative process support [16]. The cooperative agents will act on behalf of the involved roles to provide the required infrastructure for collaboration (coordination, negotiation etc.). Role-based process environments can also be used very efficiently for modelling and providing support for cooperative activities [14].

To model group processes we need to describe the relationships between the process fragments from the individual processes, cooperative activities and the roles involved. *Cooperative rules* [15] can be used to define the relationships between process fragments and cooperative activities. The cooperative rules define triggers for when the individual processes should initiate a cooperative activity, and how the individual processes should proceed dependent on the outcome of the cooperative activity. This means that there is a loose coupling between the individual process fragments and the cooperative activities making it possible to federate two different process support systems for respectively individual and group processes.

Figure 3 illustrates a process model for group processes consisting of roles, process fragments, a cooperative activity and cooperative rules (C1-C6). The cooperative rules define relationships between a cooperative activity and the process fragments for individual actors, and how a process consisting of cooperative and individual activities should be executed. Cooperative rules are typically expressed in

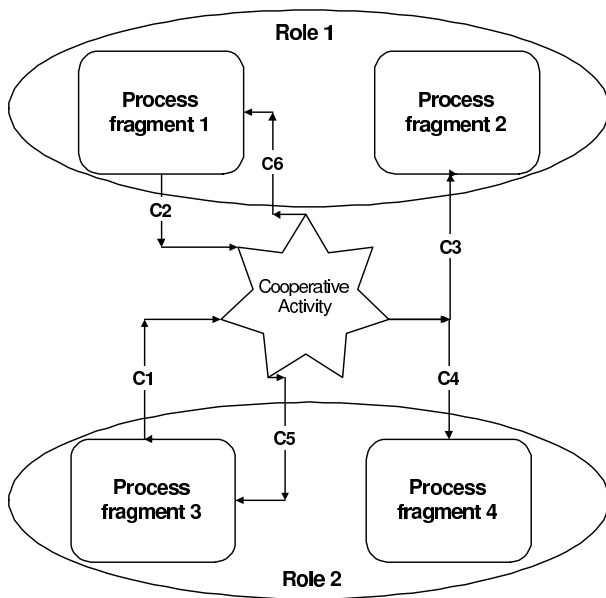


Figure 3. A group process consisting of roles, process fragments, a cooperative activity and cooperative rules.

result-reaction pairs that specify what *reactions* to be performed on process fragments based on the *results* of the outcome of a cooperative activity. Typical reactions can be to execute, move, halt, change, or add a process fragment. Reactions can in addition involve actions on cooperative activities or change the cooperative rules (reflective). More details about how cooperative rules can be specified and used can be found in [15]. In Figure 3, the cooperative rules show that process fragment 1 and 3 must be executed before the cooperative activity is executed. Depending on the outcome of the cooperative activity, process fragment 1 or 2 will be executed according to C6 or C3, and process fragment 3 or 4 will be executed according to C5 and C4. Note that process fragment 1 and 3 might need to be re-executed (iteration).

The PSE for group processes can be distributed where software developers are situated at different geographical locations.

2.3 Team Process

The team process can be characterized as high-level processes at the management level. For a manager at this level, the required process support is for following the progress of the whole project process, monitoring resource usage, performing quality assurance processes and bridging between various process models. It is not unusual at this level to have external sub-contractors that are using their own pro-

cess models and support environment. It is essential to enable an integration of the process models and support environment to enable monitoring of the whole project.

A process model at this level consists of several sub-processes that are delegated to different teams. Coordination and cooperation between groups are important aspects of the team process. The main difference between the coordination and cooperation at the group and team levels is the granularity. At the group level, the cooperation between group members is tight and frequent. At the team level, the groups are independent but the coordination is still important to e.g. manage available resources, and exchange project artifacts (document, source code, programs, tools etc.).

It is likely that different process modelling languages (PMLs) are used at the levels below (group and individual). To be able to integrate the sub-processes at the team process level, the sub-processes specify at least input and output artifacts and sub-process state. A sub-process state can be decomposed into more detailed information that can be used to determine the resource usage and progress of the group process.

Coordination activities with line management are vital to get the necessary resources. Line management will typically initiate quality assurance activities that will propagate through the different levels. The different resources have to be allocated to different groups and supervised. As the project can change, it might be necessary to reallocate resources and re-plan the whole process.

2.4 Elements of the framework

In this section we will describe the elements of the framework at the three different levels. Most of these elements are described in the previous sections. Two of the elements introduced in this section are the *Container element interface* and the *Process roles*. The container element interface specifies the required information that can be exchanged between the different levels in the process. The process roles are identified according to the role classification found in [2].

Here is a description of the various elements of the framework identified for the three levels:

Individual process

- I1 **Process modelling elements:** Activity and pre-order relationships.
- I2 **Container element:** Process fragment.
- I3 **Container element interface:** Input artifacts, Output artifacts, Process fragment state.
- I4 **Required external resources:** Tools and Artifacts.

- I5 **Process characteristics:** Activity-based, very configurable processes that can be extracted from project-planning tools.
- I6 **Required process support:** Process guidance, Activity reminders (calendar), Context-based access to information, artifacts and tools, and Automation of tasks.
- I7 **Execution environment:**¹ Desktop PCs, portable PCs, PDAs and mobile phones that can partly be connected to a LAN or wireless network.
- I8 **Process roles:** Process actor and (Sub)Process owner (personal control of individual processes).

Group process

- G1 **Process modelling elements:** Process fragment (PF), Cooperative activities, Cooperative rules and Roles.
- G2 **Container element:** Sub-process (SP).
- G3 **Container element interface:** Input artifacts, Output artifacts, Group-process state.
- G4 **Required external resources:** Cooperative tools, Configuration management and Artifacts.
- G5 **Process characteristics:** A group process consisting of several individual process fragments that are linked through cooperative activities and dynamic cooperative rules.
- G6 **Required process support:** Coordination of resources, artifacts and tools, Negotiation of resources, artifacts and tools, Synchronisation of activities, resources and artifacts, Electronic Brainstorming, Electronic voting, Collaborative authoring, and Conflict handling.
- G7 **Execution environment:** Server PCs, desktop PCs, portable PCs that are always connected through a LAN or WAN.
- G8 **Process roles:** Process Model Designer, Process Manager, and Process Actor.

Team process

- T1 **Process modelling elements:** Sub-processes, Teams.
- T2 **Container element:** Process.
- T3 **Container element interface:** Input artifacts, Output artifacts, Resources required.
- T4 **Required external resources:** Management tools and Artifacts.
- T5 **Process characteristics:** A stable process consisting of several linked sub-processes.

¹The environment the PSE is deployed to.

- T6 **Required process support:** Management-support for following progress, resource usage, initiating QA, Bridging between different PSEs.
- T7 **Execution environment:** Server PCs that are always connected through LAN or WAN.
- T8 **Process roles:** Process owner, Process Model Designer, Process manager, and Process Technology Provider.

3 The Framework Applied on XP

In this section we will relate our framework to a specific software development process, the eXtreme Programming (XP) development process as shown in Figure 4 [1]. The XP process typically starts with initial requirements that are used to produce an initial prototype of the system architecture (called an architectural spike in Figure 4). The next step in the process is to make a plan of the iterations (increments) where parts of the system are created. Every iteration starts with an activity to plan the iteration, followed by a simple design, create unit-tests, pair programming, continuous integration with the rest of the system, and unit testing. All these activities focus on one increment that usually last from a day to a couple of weeks. The last step of the process is the acceptance test with the customer. Note that the high-level activities “Release Planning” and “Acceptance Tests” are not detailed in the figure.

In the process described in Figure 4, we treat pair activities as individual activities. This is because these activities are performed in pair without requiring any extra coordination within the pair. In the XP-process we can identify the activities “Simple Design”, “Create Unit-test”, “Pair Programming”, and “Unit Testing” as individual activities. At this level the required process support is mainly to make the required tools and artifacts for doing the activity easy accessible for the developer and make it possible to add calendar entries and monitor the activity progress. In addition, the “Unit Testing” activity can be performed automatically when the “Continuous Integration” activity is finished. From the nature of the process at the individual level we can see that this process is easy to represent and understand for the developer as a network of related activities.

If we consider the *group level* of an XP development process, we can identify the cooperative activities “Release Planning”, “Iteration Planning”, “Continuous Integration”, “Stand-up meeting”, and “Acceptance Tests”. For the “Iteration” high-level activity we can identify two process fragments, where the first consists of the three connected individual activities “Simple Design”, “Create Unit-test” and “Pair Programming” and the second consists of “Unit Testing”. These two process fragments are dependent on the cooperative activities “Integration Planning” and “Continuous Integration” where more than one person is involved.

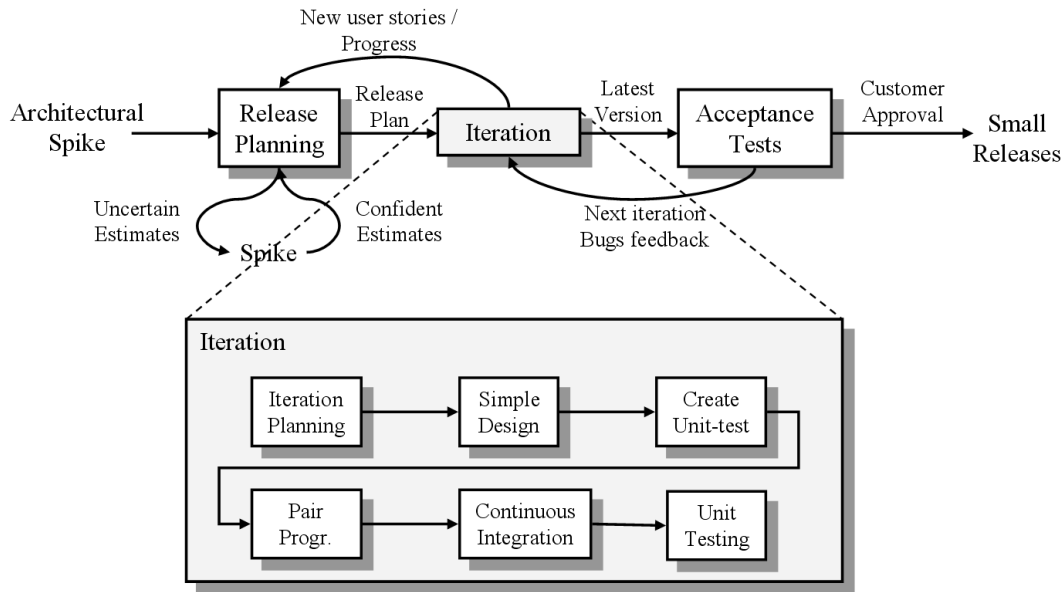


Figure 4. An overview of the eXtreme Programming (XP) development process.

The main focus at this level is interaction between various roles like in “Iteration Planning” where the stakeholders of the system must plan the next iteration. We can also recognise cooperative activities like planning and coordination of pairs (not documented in Figure 4). The pair concept in XP is very dynamic where pairs can change many times per day. The policy of how to make pairs, decides how much coordination is required of the group process. Such decisions can e.g. depend on personal skill level, expertise, free resources, and problem domain. People can quite frequently change between different process fragments making it necessary to provide context-aware process support to make these changes more optimal.

The process support should provide the infrastructure for collaboration of the stakeholders of the system. The cooperative activities are easier to represent as interacting roles or agents representing the involved parties. If we consider the two cooperative activities “Release Planning” and “Acceptance Tests”, we can identify cooperative rules that describes how the process should proceed depending on the outcome of the cooperative activities.

We consider at the team level the XP development process as consisting of the main activities “Release Planning”, “Iteration”, and “Acceptance Tests”. At this level, it is important for the project manager to have process support to follow the progress of the process. A project manager would typically like to know the status of the system and the project progress and resource usage according to the project plan etc. In Figure 4, the process of implementing an architectural prototype (spike) is not specified. A possible scenario could be that an external expert team would

create a set of architectural spikes. The external teams may have their own PSE that must be integrated at the team level. The process support system at the team level must therefore provide facilities to federate various sub-systems.

4 Related Work

The Personal Software Process (PSP) [10] was created to help software engineers doing their work well, and is based on experiences from software engineering practices that work well in daily work. Further, PSP provides detailed methods for making estimates and planning, shows how engineers can track their performance and explains how defined processes can guide their work. In PSP, the description of the software development process is only described on a high level consisting of the activities Planning, Design, Code, Compile, Test and Post-Mortem. Compared to our framework, PSP would fit to activities at the individual level. Coordination with other developers is not explicitly mentioned in PSP.

In [13], the Software Process Engineering Meta-model (SPEM) is described. SPEM is a meta-model for defining software engineering process models and their components that uses UML as a notation. A software development process is in SPEM typically described by a *Process Role* performing an *Activity* on a *Work product*. Other process model elements can in addition be expressed in SPEM like Phase, Iteration, Life Cycle, Step etc. SPEM can be used in combination with our framework to describe the software processes at different levels. However, actual enactment and

process support are not within the scope of SPEM.

[12] describes a taxonomy for characterizing meta-process categories with the main focus on process changes. The taxonomy reflects the following relevant questions to be asked concerning the meta-process: *What* aspects of the software process are allowed to be changed, due to what reasons (*Why*), *How* can the changes be implemented/installed and be propagated (*When*) by which roles (*Whom*). The *Why* aspect identifies four sources for process changes: the external world, an organization, a project environment and individuals. Our framework can be mapped to the individual (individual level) and project environment (team level). The *What* aspect identifies that three aspects of the process can be modified: Production process, meta-process and process support. Changes in the process support will affect the processes at all levels in our framework, while modification of the production process (the way software is developed) would typically be mapped to the team level. Modifications of the meta-process (changes the way to model and evolve the entire process) would typically be mapped to line support. The taxonomy described in [12] is useful for analyzing process changes at the three levels.

5 Conclusion

A common assumption for PSEs for software developing processes is to provide the same process support for all the levels in an organization. In this paper we have proposed a framework that identifies and characterise three levels in a software process. Each level has its own characteristics in terms of what PMLs should be used, what resources are used, what roles are involved and what process support that is of main concern. We believe that this framework is a useful starting point when a software development organization wants to acquire a PSE. Whether an organization wants to implement a PSE from scratch, combine existing components or purchase a complete system, it is necessary to assess the process support required and the process models involved at the different levels. In addition, our framework is useful to map different parts of a software process to the various levels to see what implications they have on the process. The framework is also useful to map existing PSEs and PMLs to the level of the organization they fit best. We hope and believe that future PSEs will provide process support dependent on the different level of organization.

Acknowledgement

This paper is a result of work in a project called MOBILE Work Across Heterogeneous Systems (MOWAHS) [11]. The MOWAHS project is sponsored by the Norwegian Research Council's IT2010 program.

References

- [1] K. Beck. *eXtreme Programming Explained: Embrace Change*. The XP series. Addison-Wesley, Reading, Mass., US, 1999.
- [2] R. Conradi, C. Fernström, A. Fuggetta, and B. Snowdon. Towards a Reference Framework for Fundamental (Software) Process Concepts. In [5], pages 3–17, Trondheim, Norway, September 7-8 1992. Springer Verlag LNCS 635.
- [3] R. Conradi, A. Fuggetta, and M. L. Jaccheri. Six theses on Software Process Research. In V. Grünh, editor, *Software Process Technology, 6th European Workshop (EWSPT'98)*, pages 100–104, Weybridge, UK, September 16-18 1998. Springer Verlag LNCS 1487.
- [4] G. Cugola and C. Ghezzi. Software Processes: A Retrospective and a Path to the Future. *SOFTWARE PROCESS – Improvement and Practice*, 4(2):101–123, 1998.
- [5] J.-C. Derniame, editor. *Proc. Second European Workshop on Software Process Technology (EWSPT'92), Trondheim, Norway. 253 p.* Springer Verlag LNCS 635, September 1992.
- [6] J.-C. Derniame, B. A. Baba, and D. Wastell, editors. *Software Process: Principles, Methodology, and Technology*. Springer Verlag LNCS 1500, Berlin, Germany, 1998.
- [7] E. di Nitto and A. Fuggetta, editors. *Process Technology: Special Issue*, volume 5 of *Journal on Automated Software Engineering*. Kluwer Academic Publisher, January 1998.
- [8] A. Finkelstein, editor. *The Future of Software Engineering*, Proc. of the 22nd International Conference on Software Engineering (ICSE'2000), Limerick, Ireland, 2000. ACM Press.
- [9] A. Fuggetta. Software Process: A Roadmap. In A. Finkelstein, editor, [8], pages 27–34, Limerick, Ireland, June 2000. ACM Press.
- [10] W. S. Humphrey. *Introduction to the Personal Software Process*. Addison-Wesley, 1997.
- [11] MOWAHS. MOBILE Work Across Heterogeneous Systems. Web: <http://www.mowahs.com>, 2001.
- [12] M. N. Nguyen and R. Conradi. Classification of Meta-processes and their Models. In *Proc. from the Third International Conference on Software Process*, pages 167–175, Washington, USA, October 10-11 1994.
- [13] OMG. Software Process Engineering Metamodel Specification, formal/2002-11-14, 2002.
- [14] A. I. Wang. An Evaluation of a Cooperative Process Support Environment. In *IASTED International Conference on Software Engineering and Applications (SEA2002)*, Cambridge, MA, USA, November 4-6 2002.
- [15] A. I. Wang, R. Conradi, and C. Liu. Integrating Workflow with Interacting Agents to support Cooperative Software Engineering. In *Proc. IASTED International Conference on Software Engineering and Applications (SEA'2000)*, pages 126–133, Las Vegas, Nevada, USA, November 6-9 2000. IASTED/ACTA Press.
- [16] A. I. Wang, C. Liu, and R. Conradi. A Multi-Agent Architecture for Cooperative Software Engineering. In *Proc. The Eleventh International Conference on Software Engineering and Knowledge Engineering (SEKE'99)*, pages 1–22, Kaiserslautern, Germany, 17-19 June 1999.