

# Differentiated Process Support for Large Software Projects

**Alf Inge Wang**

Dept. of Computer Science and Technology, Norwegian University of Science and Technology  
Sem Sælandsv. 7-9, NO-7491 Trondheim, Norway  
Phone number: +47 7359 4485, Fax number: +47 7359 4459  
Email: [alfw@idi.ntnu.no](mailto:alfw@idi.ntnu.no)

**Carl-Fredrik Sørensen**

Dept. of Computer Science and Technology, Norwegian University of Science and Technology  
Sem Sælandsv. 7-9, NO-7491 Trondheim, Norway  
Phone number: +47 7359 0731, Fax number: +47 73594459  
Email: [carlfrs@ieee.org](mailto:carlfrs@ieee.org)

# Differentiated Process Support for Large Software Projects

This chapter presents a framework for differentiated process support in large software projects. Process support can be differentiated in different levels based on the size of the development organization and the need for coordination across different levels of the organization. We have defined four main perspectives; individual, group, team, and project level, where the framework consider essential issues when planning and executing the software development processes in organizations with different levels of management. Further, a guideline is provided that suggests what is required of process support in the various organizational levels.

**Keywords:** Process Model, Process Design, Process Improvement, Project Management Methods and Tools, Process-centred Support Environments, Systems Development Process, Collaborative Support Systems, Collaborative Work Systems, Management Support Systems, Software Process Modeling, Workflow

## INTRODUCTION

Development of large and complex software systems involves large organisations. In such working environments, it is essential to plan and coordinate the process, feed the involved developers with necessary documents, tools and files, track the process and effort, and learn from and improve the process.

*Software process* modelling is aimed at understanding, guiding, coordinating, automating, and improving the software process to thus improve the software quality and reduce the effort of developing software products (Wang, 2001). Many *process models* and *process-centred support environments* (PSEs) have been created with the assumption that the same process support should be provided at every level in an organization (Conradi et al, 1998), (Derniame et al., 1998), (Nitto & Fuggetta, 1998), (Finkelstein, 2000), (Fuggetta, 2000).

If we consider development of large software systems, the organisations in such projects usually involve several levels of management. Depending on the level of an organisation a person is working in, the perspective and goal of the work will vary. For a programmer, the main concern would be to have access to all necessary files, documents, and tools to carry out efficient programming. Personnel working at higher levels in the organisation would typically have other concerns like coordinating people, scheduling of the process, quality assurance, planning of activities etc. Thus, it is essential that the process support in such organisations reflects the levels being supported. It is also important that the way the processes are modelled is tailored for the organisational level and the characteristics of this level.

This chapter presents a differentiated process support framework that describes the elements required to model the software process, the required external resources (like tools and documents), and the required process support provided by a process-centred environment. Our framework describes the required process support from four perspectives: At the individual level, at the group level, at the team level, and at the project level. Thus, the objectives of this chapter is to give insights into essential issues to be considered when planning and executing a software development process for large software projects consisting of several levels of management. The chapter also provides a guideline for what is required of process support for

the various levels of an organisation. This guideline can be used as input when evaluating tools to be used to support the development and management processes of large software projects.

## BACKGROUND

This section gives an introduction to the background and important terms used in our framework, and describes related work.

### ***Software Process and Software Process Modelling***

At a NATO Conference in Garmisch-Partenkirchen in 1968, the term *software engineering* was first introduced (Naur & Randell, 1969). The conference discussed what was called the “software crisis” introduced by third generation computer hardware. The work within the software engineering domain has been concerned with methods, techniques, tools, programming languages, and more to face the problems in software projects like delayed products, cost overruns, and bad reliability and performance in software products. Despite the many efforts to try to solve the software crisis, we are still struggling with the same problems as they did in the sixties. Brooks (1986) argues that there is no “silver bullet” that can solve all problems in software engineering. The only way to limit the negative effects identified as the software crisis is to use best practices in the many areas of software engineering. There are no best practices that solve all problems, but in combination many issues can be eliminated. One best practice is to improve the software process itself involving all the activities necessary in developing a software product. This chapter is intended as a guideline for improving the software process at different levels in a software development organisation.

Before we take a closer look at the software process support for different levels of an organisation, it is necessary to agree on the central terminology used in this chapter. As mentioned before, the term *software engineering* covers most aspects involved when developing large software systems. According to Sommerville (1995):

*“Software engineering is concerned with software systems which are built by teams rather than individual programmers, uses engineering principles in the development of these systems, and is made up of both technical and non-technical aspects”.*

As Sommerville states, software development involves more than the technical aspects like dealing with the source code of the system. Important non-technical aspects of developing software involve scheduling, budgeting, resource management, etc. In addition, the term *software process* (Lehman & Belady, 1985) is used to denote all activities performed within a software organisation.

In this perspective, *software process modelling* describes the activities of understanding and describing a given software process (Wang, 2001). There has been proposed several elements that need to be identified to describe a software process, but the most common model involves the elements *products* (a set of artefacts related to the software product), *resources* (assets, like tools or human resources needed to carry out the process), *activities* (steps of the process) and *directions* (policies, rules, and procedures that govern the activities) (Derniame et al.,

1998). To be able to provide process support, the software process must be modelled. Software process modelling is according to Høydalsvik (1997) defined as:

*“The activities performed, and language and tools applied, in order to create models of software development processes within a software development organisation”.*

Another term related to software process is **workflow**. Workflow is by the Workflow Management Coalition (1999) defined as:

*“The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules”.*

The main difference between software process and workflow is that workflow focuses on business processes. However, this does not rule out the possibility to use workflow tools to support the software process.

When talking about software processes, we also often mention the term **software lifecycle** (Royce, 1987) that describes the life of a product from the initial stage to the final production and maintenance. A software lifecycle consists of several phases that a software product usually must go through: requirement specification, system design, implementation, testing and operation, and maintenance. During these phases, the required process support will vary. For the two first phases, the main focus is on analysis and documentation. For the three last phases, the main focus is on the source code and computer system. For all these phases, various tools exist to make the software development easier.

In addition, there are other parts of the software process that is phase independent, e.g., software configuration management, software cost estimation, software quality management, software process improvement, and software process technology (Wang, 2001).

The lifecycle of a software product can be organised in various ways. The classical approach is the *Waterfall model* (Royce, 1987) that describes the lifecycle as a predefined sequence of phases, where each phase has a set of defined activities. Each phase has an input and output, and the output from one phase is the input to the next. The Waterfall process is in principle linear and a succeeding phase should not start until the previous phase has been finished. The *evolutionary development model* is another way of viewing the software development process. This approach interleaves the phases; specification, implementation and validation, and is based on rapid prototyping. A prototype is developed early in the project in cooperation with the customer and is refined until the customer is satisfied. An example of an evolutionary development model is the *Spiral Model* (Boehm, 1988) that can incorporate different development processes through separate iterations.

*Agile software development* models like *eXtreme Programming* (Beck, 1999) focus on being lightweight and flexible. Such model embraces short development iterations to be able to cope with up-coming changes in requirements, minimum project documentation apart from comments in source code, user involvement during development, and other best-practices. Agile methodologies fit very well for high-risk projects with unstable requirements.

A software process model gives an overview of all the required activities and how they relate to each other, documents, tools, actors, methods, techniques, standards etc. This chapter is not

limited to process support of a specific lifecycle model, but focuses rather on the required process support independent of how the software process is modelled and carried out in detail. Thus, this chapter describes guidelines that are important to incorporate when planning and supporting a software process. This may make it easier to improve the software process, thus improving the software product.

### ***Related Work***

The framework described in this chapter divides the software process into four distinct levels. The Personal Software Process (PSP) (Humphrey, 1997) defines the software process at the individual level in a software development process. The intension of PSP was to help software engineers to do their daily work well by applying practices that work well based on experiences from the software engineering field. Further, PSP describes detailed methods for making estimates and plans, shows how software engineers can track their performance, and describes how defined processes can guide their work. The process model in PSP is described only at a high-level. The model consists of the activities Planning, Design, Code, Compile, Test, and Post-Mortem. The PSP fits very well with the individual process level described in this chapter. However, note that PSP does not say anything explicitly about coordination of developers and project management on a higher organisational level.

The Software Process Engineering Meta-model (SPEM) defines software process models and their components using UML notation (OMG, 2002). In SPEM, a software development process is usually described by a *Process role* performing an *Activity* on a *Work product*. In addition, SPEM provides modelling elements to express phases, iterations, lifecycles, steps etc. This means that SPEM can visualise different organisational levels of a software process. Thus, SPEM can be used in combination with our framework to describe the software processes at different levels. However, SPEM was not made with software process enactment in mind. This means that to provide process support for a process specified using SPEM, the model has to be translated to an enactable process model language (PML).

UML4SPM is executable software PML based on UML 2.0 (Bendraou et al., 2006). In UML4SPM PML, a software process can be represented at different hierarchical levels by using the main building blocks Process, Activity, and Action. The organisational structure can also be modelled in a hierarchical manner using the model building blocks Team and Agents. Although UML4SPM support modelling of a software process at different levels, it uses the same PML to represent all levels of an organisation. As the PML represents processes as activity-networks, collaborative activities beyond coordination are hard to represent in such PMLs (Wang, 2002).

Serendipity-II is a process management environment that supports distributed process modelling and enactment (Grundy et. al, 1998). The PML allows hierarchical modelling of software processes represented as process stages and sub-processes. Moreover, Serendipity-II is based on a decentralized distributed architecture that enables users to collaboratively edit process models synchronously and asynchronously. This approach enables support for autonomous sub-organisations that can tailor their specific part of the process to their needs, which is essential to support different levels of a software organisation.

Kivisto (1999) describes the roles of developers as a part of a software process model for development of client-server process. Kivisto proposes a process model in three dimensions:

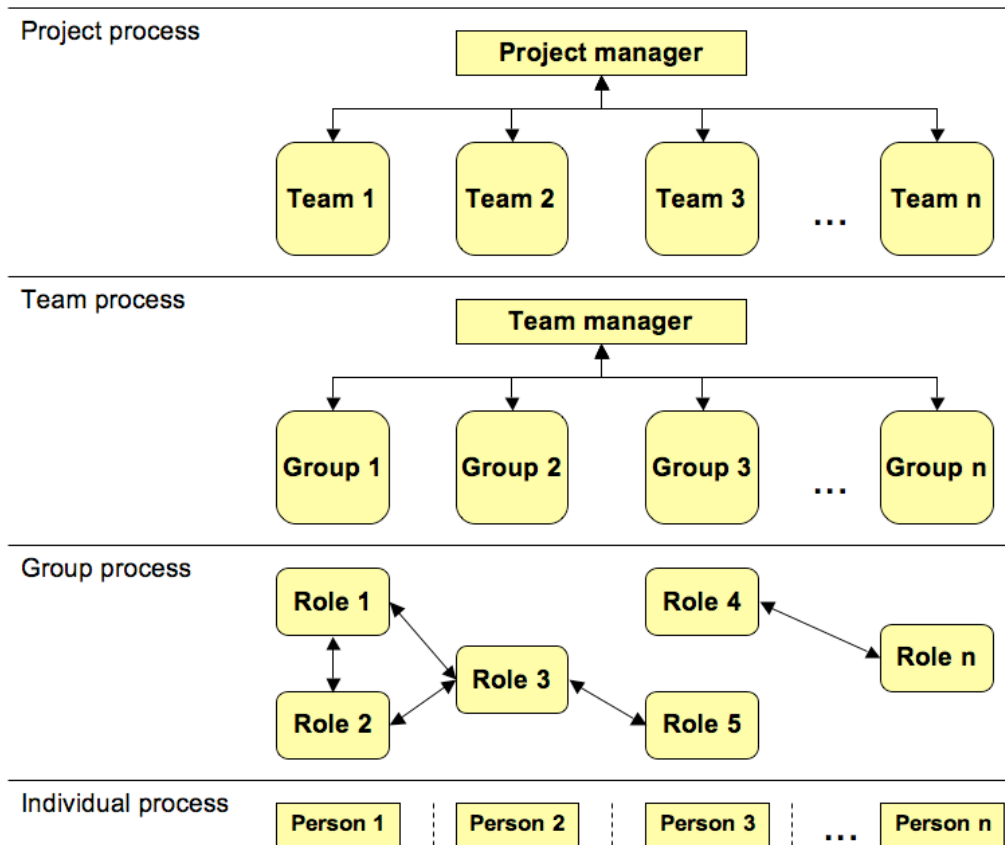
Organisational, Technological, and Process. The organisational dimension focuses on what roles are involved in the development process and how these roles are organised. The technological dimension focuses on issues related to client-server software architecture and technology choices for such system. The process dimension gives an overview of how the development process can be organised in phases and tasks, and the assignment of roles to specific tasks.

A taxonomy for characterising meta-process categories with the main focus on process changes is presented in (Nguyen & Conradi, 1994). The taxonomy reflects the following 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 organisation, a project environment, and individuals. The framework described in this chapter can be mapped to the individual (individual level), project environment (group and team level), and organisation (project level). Changes in the process support will affect the processes at all levels in our framework, while modifications of the production process (the way the software is developed) would typically be mapped to the team level and below. Modifications of the meta-process (changes and evolution of the process model) would typically be mapped into the project level. The taxonomy described in (Nguyen & Conradi, 1994) is useful for analysing process changes at all four levels in our framework.

## **PROCESS SUPPORT IN DIFFERENT LEVELS IN A SOFTWARE DEVELOPMENT ORGANISATION**

This section describes issues that must be considered when planning the process support for large software projects being organised in groups and teams (typically 20 or more people). In such software development organisations, the process support required is very dependent on where you are in the hierarchy in the organisation. E.g., the software process support needs for an individual developer is totally different from the needs of a project manager.

Our framework as detailed below, describes the software process support from four perspectives based on the organisational level as shown in Figure 1. The *first level* in the framework focuses on the *individual process* reflecting the personal software process in the daily work. The *second level* describes group processes where two or more individual collaborate and interact. The *third level* is concerned with management of groups within a team. Finally, the *fourth level* is concerned with project management where several teams are involved. Typically for smaller projects, the team and group levels will merge.



**Figure 1 Overview of process support in different levels**

Table 1 shows the four levels of process support required in a big software organisation. For each level, the model describes:

- **Process elements:** The required process elements needed to model the process.
- **Process relations:** The required relationships between process elements.
- **Required external resources:** The resources used by the process-centred support environment (PSE).
- **Required process support:** The process support required at a specific organisational level.

Org. level	Process elements	Process relations	Required external resources	Required process support
	<i>Interface: Product, Experiences and knowledge, Resources, Project management rules and practice</i>			
Project process	Teams, Team processes, Milestones	Team coordination, Assigned team(s)	Project management tools and Artefacts, Experience-/ knowledgebase, Estimation models and tools, Resource tools,	Project planning, Project management, Experience exchange, Project estimation, Resource management
	<i>Interface: Input and output artefacts, Team and resources assigned, Team process state</i>			
Team process	Groups, Group processes	Group coordination, Assigned group(s)	Team management tools and Artefacts	Team management
	<i>Interface: Input and output artefacts, Group assigned, Group process state</i>			
Group process	Roles, Process fragments, Cooperative activities	Cooperation and coordination rules, Assigned role(s)	Cooperative tools, Configuration management and Artefacts	Coordination, Negotiation, Collaboration
	<i>Interface: Input and output artefacts, Role assigned, Process fragment state</i>			
Individual process	Activities	Assigned activities	Tools and Artefacts	Process guidance, automation, calendar

**Table 1 Process support in an organisation**

The framework specifies interfaces that describe the input and output between levels. The uppermost interface specifies the relationships to the company and customer(s). Some activities are carried out at all levels of an organisation, e.g., quality assurance and configuration management.

The rest of this section will describe each level of the framework in more detail.

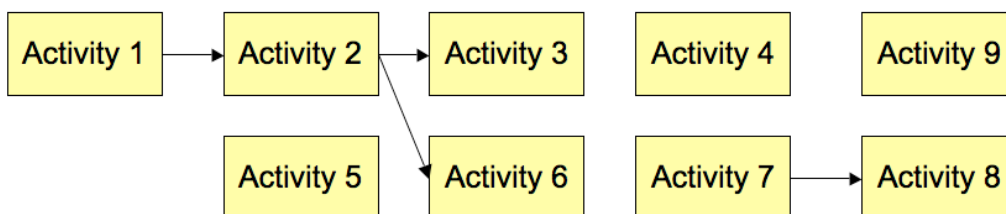
### ***Individual Process***

The lowest process level in an organisation is the *individual* process level focusing on the individual tasks of a software developer. An example of a role related to this level is a programmer with typical tasks like reading documents, writing code, writing documentation, debugging code, compiling code, building code, etc. Generally, the process of an individual actor typically consists of a set of activities related to the roles that the actor plays (an actor can play more than one role). It is very important that the process support is adaptable and configurable, making it possible for the actor to fit the work to personal preferences. The required process support is very dependent on the experience level of the individual actor. An inexperienced person would typically need process guidance to show the tasks 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 a hindrance to work effectively. For the latter, it would be more useful to provide automation of

repetitive tasks. Independently of the experience level of the actors, the process support should provide quick and easy access to the required tools and artefacts. The activities of the individual process should be integrated with personal digital calendars to enable activity states and deadlines to be accessible on personal computers, personal data assistants (PDAs), and mobile phones. This means that the execution environment for the process support also must be capable of communicating and utilising mobile devices and networks.

The framework presented in Table 1 identified activity as the main building block for individual process models. Another approach could be to use roles with responsibilities as the main building block. However, the most common approach is that individuals represent the daily work as a set of activities. Also, the most used process tools for individual processes also focus on activities, like calendar software. In addition, activities or tasks are the most common output of many project-planning tools, where individuals are assigned a set of activities. This means that using activities as process model elements makes it easier to integrate with higher-level management processes and tools. In addition, since people are used to the representation of work as a set of activities and tools that represent the work as activities, it is also easier for the individuals to tailor their own processes (reschedule activities etc.). In some cases, it can be useful to let an actor model her or his own processes. In (Wang, 2002), an experiment indicates that it is easier to use activity-based than role-based modelling for individual processes. This is especially true for inexperienced process modellers.

The process relation with respect to individual processes is the assign relation. A team or project manager typically assigns activities to an individual actor. In some cases, some activities must be carried out in a specific sequence putting constraints on how the work should be carried out and how the individual process can be modified. A representation of an individual process is illustrated in Figure 2.



**Figure 2 An individual process model**

The process model in Figure 2 shows a collection of activities. Some of the activities have pre-order relations (the activities 1, 2, 3, 6, 7, and 8). The remaining activities have no constraints and can be carried out regardless of the state of the process.

In this chapter, we define a collection of related activities (e.g., activities related to the same phase in the process and the same project) to be a *process fragment* (PF). The name process fragment indicates that this collection of activities is part of a larger process at a higher level of the organisation. The process of assigning activities to an individual actor in the organisation is carried out by delegating one or more process fragments to the actor. The state of the process fragment is determined by looking at the state of the activities. A process fragment is completed when all activities in the fragment are completed.

The goal of the individual process is to produce various artefacts like requirement and design documents, source files, build files, test files etc. To provide a sufficient software process support at this level of the organisation, it is essential that the PSE can be integrated with production tools such as office applications, integrated development environments (IDEs), and other similar tools. The process support environment should make it easy to access all necessary tools and files, and handle configuration management.

When the individual actor initialises her/his process, the process models could either be fully specified by, e.g., a team or project manager or be provided as a process template where all necessary activities are defined. For the latter, the actor must detail her/his own process to meet her/his needs and preferences. During the process enactment, the individual actor might also add, change, or delete activities depending on how the process proceeds (e.g., change of project plan, changed priorities, or changed process goals). Some PSEs might provide support for changing the process automatically or semi-automatically. It is important that the PSE is flexible enough to managing instantiation of template process fragments as well as allowing changes of the process during process enactment (Grundy et. al, 1998).

### **Group Process**

The next level in the framework is the *group* process level focusing at the cooperative aspects of the software process. Traditionally, software process modelling and enactment of software processes have been focusing on "forcing" and guiding people to work according to a specified model, where interaction between people has been coordinated through a strictly defined control/data flow. Such approaches are usually based on modelling processes as activity-networks consisting of sub-processes, activities, tasks, or operations. Cooperative aspects of the software development process have often been either eliminated or ignored because it has been hard to model cooperative activities in existing systems, or there has not been an interest for doing so. The software development processes are also human-centred processes. Cugola and Ghezzi (1998) state that "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".

The cooperative process involves coordination of activities carried out by several actors or roles, cooperative activities where two or more persons must participate to complete the activity, and coordination and negotiation of artefacts and resources.

At this level, **configuration management** (CM) is very important since two or more actors often will share the same resources (including files). The sharing of resources might cause conflicts where two or more actors would like to update the same data at the same time. Thus, the CM environment must be integrated with the PSE to provide support for initiating negotiation processes in case of conflicts regarding resources (e.g., files), and support for synchronising output artefacts from the process (Wang et al., 1998) (Wang, 2000).

The group process defines the synchronisation points for individual processes involving several actors. These synchronisation points represent parts of the software process where files or other resources need to be exchanged and synchronised (mainly coordination of artefact and process). In addition, the group process involves cooperative activities where two or more actors are involved. Example of such activities can be distributed brainstorming,

electronic voting, collaborative authoring, and conflict management. Cooperative activities have very different characteristics compared to individual activities. While the main emphasis of the individual activities is on the activities themselves, cooperative activities are all about interaction and coordination between roles. This means that the process support for cooperative activities must provide an infrastructure to enable efficient interaction of the involved roles and to enable flexible exchange of artefacts. Note that most people working in a large software project will be involved in both individual and group processes.

The cooperative processes can be represented in many ways. However, to represent cooperative processes as a set of related activities, as described in previous section, does not make sense since the focus is on interaction between roles. We propose two alternative approaches to represent cooperative processes at the group level:

### **Cooperative software agents**

Cooperative software agents can be used to represent actors involved in cooperative activities. The cooperative agents will act on behalf of the involved actors to provide the required infrastructure for collaboration (coordination, negotiation, etc.). In (Wang et al, 2000), a cooperative agent framework developed to provide process support is presented. In this framework, every actor has her/his own workspace where she/he can interact and configure her/his agents. Agents interact on behalf of various actors in agent meeting places that represent neutral workspaces where services and artefacts can be exchanged. The collaborative aspects are supported through coordination agents, negotiation agents, and mediation agents. When software agents are used, the agent environment usually provides a set of ready-to-use agents that provide support for various collaborative tasks and that can be configured for specific user needs. If the required functionality is not supported by the predefined agents, new agents can be implemented using a high-level API. A similar approach to support group processes using software agents is described in (Glaser & Derniame, 1998). This approach uses agents to look for competence profiles that match the tasks to be carried out.

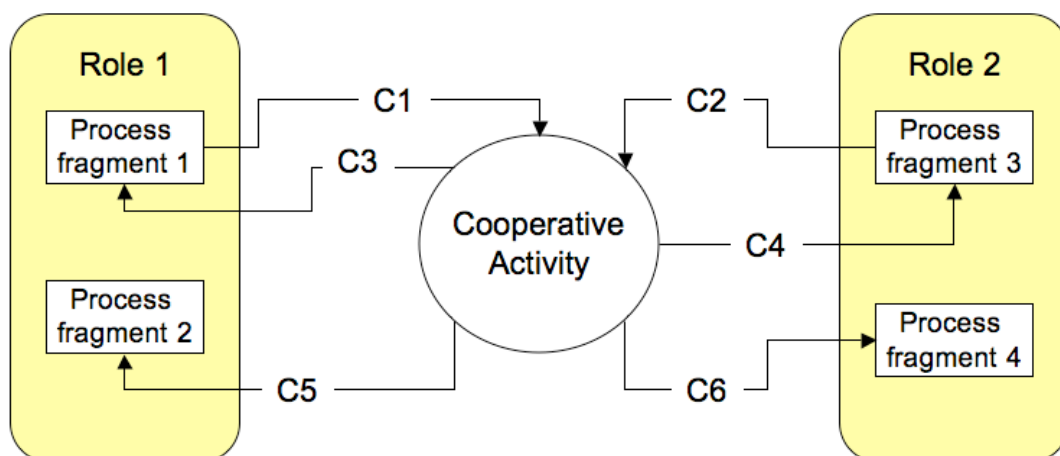
### **Role-based process environments**

Role-based process environments can also be used very efficiently to model and provide support for cooperative activities, e.g., like the workflow system ProcessWeb (Yeomans, 1996). In ProcessWeb, all involved in the process are modelled as *roles*, and *interactions* are used to provide communication channels between roles. A role is defined by its *actions* (methods) and its *resources* (attributes). Preconditions called “*when guards*” are used to select action for enactment, and they are expressed as **if** statements. Interactions are uni-directional, asynchronously typed communication channels provided through a *takeport* and a *giveport*. A *takeport* receives data or control flow from another role, and a *giveport* sends data or control flow to another role. Similar approaches can be found in (Fadlia et. al, 2005). Most role-based process modelling approaches are object-oriented and the modelling process will be similar to programming in an object-oriented programming language. This gives a very flexible and expressive PML, but it requires a certain level of expertise to handle.

A comparison how well cooperative activities can be modelled and supported in software agents, activity networks, and role-based process environments is described in (Wang, 2002). The results show that activity networks are not well suited, but software agents and role-based process environments are able to represent cooperative activities very well.

In the previous section, the term process fragment was used to denote a collection of activities that represents the process for an individual actor. To model group processes, it is necessary to model the relationships between the process fragments of the individual actors as well as cooperative activities and the roles involved. *Cooperative rules* (Wang et al., 2000) can be used to define the relationships between process fragments and cooperative activities. The cooperative rules specify conditions for cooperative activities that should be initiated depending on the condition of the individual process fragments. The cooperative rules also specify how the individual processes should proceed depending on the outcome of a cooperative activity. Thus, the cooperative rules specify a loose coupling between the individual process fragments and the cooperative activities enabling federation of different PSEs for individual and group processes.

Figure 3 illustrates a group process model consisting of roles, process fragments, a cooperative activity, and cooperative rules (C1-C6). The model describes how cooperative rules describe the relationships between process fragments (individual processes) for each role and the cooperative activity. The cooperative rules are modelled as result-reaction pairs that specify the *reaction* that should be taken based on the *result* (outcome) of a cooperative activity or a process fragment. Typical reactions can be to execute, re-group, halt, change, or add a process fragment. In addition, a reaction can initiate or change cooperative activities (e.g., initiate a specified agent) or change the cooperative rules (reflective).



**Figure 3 Group process model**

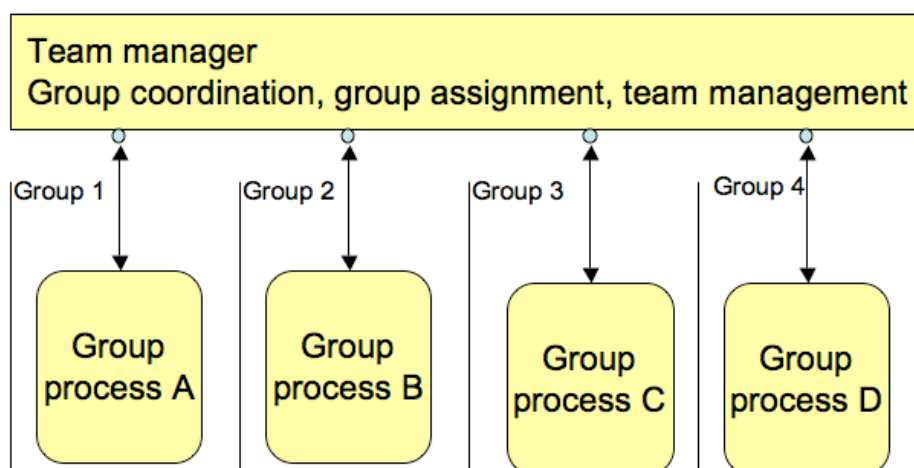
The cooperative rules in the process model in Figure 3, specifies that *Process fragment 1* and *3* must be completed before the cooperative activity will be executed (C1 and C2). Depending on the outcome of the cooperative activity, either *Process fragment 1* and *3* will be re-enacted (C3 and C4), or *Process fragment 2* and *4* will be initiated (C5 and C6).

In addition to provide support for cooperative activities in a PSE, it is also important for the PSE to provide access to other collaborative tools like chat-applications, collaborative editors, Wikis, discussion forums etc. The output of group processes is artefacts produced by the group and the individuals.

## Team Process

The *team* process level is related to middle management and specific roles in a software project like a team manager. The team manager manages the development of one or more main parts of a software system within the competence of the team (e.g., security, business logic, etc.). Typical tasks at this level can be planning of group processes, assignment of group process to groups, tracking progress of team, monitoring resource usage, and low-level quality assurance. The process support required at this level is monitoring of the on-going processes, computation of resource usage, prediction and estimation of progress and resource usage, and creation and editing of process models. It is also essential that the team process model can integrate the group and individual process models and provide a process environment that will exchange artefacts and state changes between the individual and group processes. 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 artefacts (document, source code, programs, tools etc.). It is the team manager who is responsible for the coordination of groups. Thus, it is important that the process support provides the infrastructure and tool to support and ease this job.

Figure 4 illustrates a process model at the team level that consists of team process coordination and assigning group processes in addition to other team management activities. The group processes are coordinated through defined interfaces provided by the PSE (illustrated by the small circles). Another alternative could be to allow direct coordination between groups. This approach might cause problems since two levels of coordination must then take place at the group level: coordination of individuals and coordination of groups. Another advantage by coordination groups through a team manager is that it is easier to make correct decisions about changes of the process, as this level gives a better overview of the process and the state of the different parts of the process.



**Figure 4 Team process model**

The output from the team process level is the artefacts from all groups as well as the state of the team process.

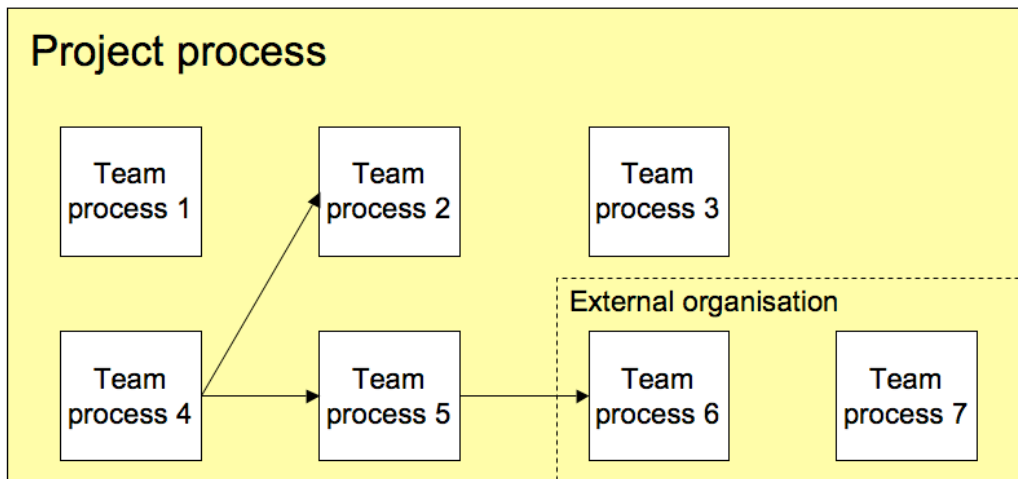
## **Project Process**

The *project* process level represents the software development process at the highest abstraction level, usually managed by a project manager. For this role, the required process support involves tools and procedures to create a project plan (project process model) for the whole project, establish and change teams and groups, assign personnel to teams and groups, estimate project resource usage, track progress of the whole project, monitor resource usage, perform company quality assurance procedures, assess and improve the development process, etc.

At this level it is not unusual to have external sub-contractors to carry out parts of the project that require special expertise or expertise that is not present in the company. From a process support point of view, it is also likely that external sub-contractors use their own process models and support environment. Thus, it is essential for the process-centred support environment to enable federation of various process model languages and tools to enable monitoring and process enactment of the whole project.

Another essential activity at the project level is to package artefacts, process models, and knowledge and experiences from completed projects that can be used as input when planning, estimating, modelling, and managing new projects. One method for harvesting experiences from completed projects is to use post-mortem analysis methods (Birk et al., 2002) like Affinity Diagram (KJ diagrams) (Scupin, 1997) and Root Cause Analysis (Straker, 1995). Affinity diagrams provide a method for carrying out a structured brainstorming focusing on successes and problems in the project. The root cause analysis investigates one particular issue found in the brainstorming process (positive or negative) to discover the causes and sub-causes of the success or problem. In addition, reusable code and documents need to be identified and described with meta-information to enable search on historical information. A knowledgebase or experience-base is the most common tool to store and manage project experiences and reusable artefacts (Dingsøy & Røyrvik, 2003).

Figure 5 illustrates a process model at the project level. A project process consists of several team processes. Some of these team processes must be carried out in a specific order (e.g., Team process 4, 2, 5, and 6). The pre-order constraints are most often caused by one team process that requires a finished product from another team process before it can start. Note also that parts of the process might be carried out by external organisations. This might implicate that this part of the process model will be expressed in a different process modelling language compared to the in-company models. Also this part of the process model might need integration with another PSE.



**Figure 5 Project process model**

The output of the project process is the final software product along with experiences and knowledge acquired during the project. The completion of a project will also release resources that can be used in new projects. The input to the project process should also be experiences and knowledge from previous projects. Further, a new project is allocated resources like people, tools, equipment, and money. Other important inputs to the project process are company policies and project management rules and practices that must be followed. These constraints will influence the way the process is organised and how the software process will be modelled.

### ***Organisational Issues***

This section has so far only briefly mentioned the organisational issues that have to be taken into consideration when planning multi-level process support. This section will briefly look at organisational issues that must be taken into account.

At all levels of an organisation, people involved play different roles related to the activities they are involved in. What roles people play varies from project to project. The role also depends on the domain of the project, the company culture, and the culture of the nationalities of the people employed in the company. Kivisto (1999) describes the roles involved in the development of client-server applications to be: application developer, business object developer, database developer, end-user, team leader, project manager, quality assurer, tester, and expert. Further, these roles are mapped into a hierarchy where the project manager is responsible for managing team leaders, and team leaders are responsible for managing the rest of the roles. This organisational decomposition fits well with the two top levels and the bottom level of our framework. For a specific project, the roles involved must be described and people that fit these roles must be found. An employee profile database can be used for this purpose to find the correct person for a specified role.

McGrath (1998) has proposed a software process modelling framework that incorporates the behavioural issues of software development. The framework has three levels: the universe of discourse (level 1), the conceptual model (level 2) and the external models (level 3). The framework includes behaviour aspects such as attitudes, opinions, beliefs, knowledge,

organisational power, politics, culture, and structure. These important issues are not covered in the approach presented in this chapter.

## **CREATION OF A MULTI-LEVEL PROCESS MODEL**

We propose two main approaches to create a multi-level process model for a software project: top-down and bottom-up. These two approaches are presented in this section.

### ***Top-down Approach***

In the top-down approach, a project manager will typically first create a high level process model based on experiences from previous projects found in the knowledgebase, and the present information (constraints and resources) about the new project. This process model should at least identify the processes at the team process level. Further, the model might contain representations of group and individual processes at a coarse level. It is important that the individual and group processes are not constrained in such a way that they cannot be tailored by the process participants carrying out the processes. If the group and individual processes are modelled by a project manager (at the project level), then the process model should be described by a process template rather than an instantiated process model. If the project manager has not modelled any group or individual processes, the team manager must identify and model individual and group processes. It is also essential that the individual activities are not modelled in such a way that they cannot be altered or tailored by the process participants at the individual process level. The team manager should include sufficient content to the individual process level like activity description, necessary documents and tools, and a proposed sequence of the activities. In some cases the sequence of activities must be frozen because of coordination and synchronisation between developers. The top-down approach is suitable when the top management has a good idea of how the software is being developed in the company or the company management wants to enforce a specific way of developing software motivated by software improvement initiatives (e.g. CMM). For the latter, it is important that the project manager has a good dialogue with the people working in the teams to avoid sabotage of the process. If a process is enforced on the individual actors, they need to be motivated and know why they have to do their work in a particular way (e.g., because of quality or economical reasons).

### ***Bottom-up Approach***

The bottom-up approach aims to harvest the process models by starting at the individual processes and moving upwards in hierarchy of the software organisation. This means that the team process models will be created based on the underlying group and individual process models, and that the project process model will be created based on the underlying team process models. If no processes or process models are described or defined in the company, the bottom-up approach can be applied to create a process model that represents how the organisation is currently working. This process model can be used as a starting point to improve the process by identifying parts of the process that are cumbersome or uses unnecessary resources. However, it is likely that the process needs to be simplified to be able to model it and provide process support for it.

## **What Approach to Choose?**

There is no clear answer if a software organisation should use the top-down or the bottom-up approach. The choice depends on the strategic goals of the organisation and on the working routines used. If the current work processes are causing several problems, a top-down approach inspired by established software development practices could be used to establish a new and more efficient way of working. If the current work processes are efficient and the employees are pleased with the way they work, a bottom-up approach should be chosen. Many organisations are in between these two extremes. If this is the case, a combination of top-down and bottom-up can be used. To succeed in combining the two approaches, it is necessary for the different levels of the organisation to interact often and synchronise the process models to make sure that they fit. This chapter can be used as a checklist of what should be considered at each level of the organisation. In addition, the chapter identifies the interfaces between the organisational levels making it possible to provide one process model for the whole project consisting of autonomous parts.

## **CONCLUSION**

This chapter has presented a framework that describes process support in four levels in a large software development organisation.

The *first level* is the individual level where the main focus is on the activities of the individual developer and the process support she/he requires. At this level it is important that the process support can be integrated with calendar applications running on personal computers, personal data assistants, or mobile phones. In addition, it is essential that the process support provides easy access to the required files and tools.

The *second level* is the group level where actors involved in the development process need to collaborate, and coordinate and negotiate shared resources. The process model at this level focuses on cooperative activities, cooperative rules, and the process fragments of the individual participants. The cooperative activities can be modelled and supported using software cooperative agents or role models. Since files are likely to be shared at this level, configuration management is essential. The configuration management system should be integrated to the process-centred support environment to provide process support for negotiation for shared resources.

The *third level* is the team level where the team manager coordinates the various groups in the team. The main focus at this level is team management that involves assigning work to groups, and provide the necessary infrastructure and support for inter-group coordination.

The *fourth level* is the project level where the project manager initiates, manages, and finalises projects. At the end of a project, it is important that experiences, knowledge and reusable artefacts are packaged in a knowledgebase for later usage. In the initialisation phase of a new project, the knowledgebase plays a vital role when making resource estimates, modelling the process, assigning resources, and finding the reusable software components. It is also important at this level to make a heterogeneous PSE able to interact with PSEs in external organisations, as some team processes might be carried out by external resources. Finally,

support for monitoring the process consisting of several teams that also might be external to the company is essential.

The main contribution of this chapter is a description of the process model elements required in a process model at each level of a software development organisation, and a description and discussion about the required process support at these levels.

## **FUTURE RESEARCH DIRECTIONS**

Professional industrial software development organisations are always looking for ways to improve their development and project processes. One way of improving the software process is to provide the appropriate support for the process at different levels of the organisation. Most research within software process modelling assumes software development made by one company with a stable organisation and where the requirements are stable. This is not always the case.

The success of large open source projects have made open source development practices interesting to also apply within larger software organisations that perform global software development.

The ITEA COSI project (Co-development using inner & Open source in Software Intensive products, <http://itea-cosi.org/>) has observed a shift in software development, to more cooperation, new coalitions, and more collaboration with open source communities. These forms of network-enabled collaborations create new challenges for software developers (Lacotte, 2004). These new forms of development makes the normal development processes less scalable and flexible since more and more of the software is harvested through reuse of internal components, through COTS, and/or through open source projects. In addition, a lot of the actual development efforts are out-sourced, making it harder to ensure quality, productivity, and control of the software project(s).

The percentage of code actually produced locally in an organisation is decreasing, while the complexity and size of the products are still increasing. Large software companies like IBM and Sun Microsystems, have made parts of their product portfolio as open source, creating precedence for how to make and publish software.

Participants in open source projects are increasingly employed by industrial companies. Many of these participants are very flexible with respect to how software is developed, and will often adapt the development practices used in the open source project.

When considering software process support for open source processes, the most important issues that must be considered are collaboration and coordination between many distributed developers and flexible and open PSEs that can be integrated with a large set of tools.

Agile methods (Abrahamsson et al., 2003) have gained much attention the recent years by moving the attention from documents and specifications, to customer participation and incremental and iterative development of the products. Smaller projects are more likely to adopt agile methods, while large projects still will employ project management practices that are more heavy-weight. To use agile methods in larger organisation, the main problem is to manage multiple cooperating teams that facilitate various overlapping, informal cross-team communication. Kahkonen (2004) describes an approach for introducing agile methods in large organisation is presented. This approach was tested by Nokia with success, where the

informal cross-team communication problem was solved by using communities of practices theory.

Our proposed framework is embracing these new trends by differentiating the process model and support for different levels of organisation.

## REFERENCES

Abrahamsson, P., Warsta, J., Siponen, M. T., & Ronkainen, J. (2003), *New Directions on Agile Methods: A Comparative Analysis*. 25<sup>th</sup> International Conference on Software Engineering (ICSE'03), Portland, Oregon, USA.

Beck, K. (1999). *Extreme Programming Explained: Embrace Change*. Addison-Wesley, Reading, MA.

Bendraou, R., Gervais, M-P. & Blanc, X. (2006). *UMLASPM: An Executable Software Process Modeling Language Providing High-Level Abstracts*. 10<sup>th</sup> IEEE International Enterprise Distributed Object Computing Conference (EDOC'06), Hong Kong, China.

Birk, A., Dingsøyr, T. & Stålhane, T. (2002). *Postmortem: Never Leave a Project without It*. IEEE Software, vol. 19, no. 3, May/Jun.

Boehm, B. W. (1988). *A Spiral Model of Software Development and Enhancement*. IEEE Computer. May.

Brooks, F.P. (1986). *No Silver Bullet: Essence and Accidents of Software Engineering*. Information Processing'86. North-Holland, IFIP.

Conradi, R., Fuggetta, A., & Jaccheri, M., L. (1998) *Six theses on Software Process Research*. Software Process Technology, 6th European Workshop (EWSPT'98), Weybridge. Springer-Verlag. LNCS 1487.

Cugola, G. & Ghezzi, C. (1998). *Software Processes: a Retrospective and a Path to the Future*. SOFTWARE PROCESS – Improvement and Practice, 4(2).

Derniame, J-C., Baba, B.A., & Wastell (Eds.) (1998). *Software Process: Principles, Methodology, and Technology*. Springer Verlag LNCS 1500, Berlin, Germany.

Dingsøyr, T. & Røyrvik, E. (2003). *An Empirical Study of an Informal Knowledge Repository in a Medium-Sized Software Consulting Company*. 25<sup>th</sup> International Conference on Software Engineering (ICSE'03), Portland, Oregon, USA.

Fadila, A., Said, G. & Nora, B. (2005). *Software Process Modeling Using Role and Coordination*. Journal of Computer Science, 2(4).

Finkelstein, A. (Ed.) (2000). *The Future of Software Engineering*. 22nd International Conference on Software Engineering (ICSE'2000), Limerick, Ireland.

- Fuggetta , A. (2000). *Software Process: A Roadmap*. In A. Finkelstein, editor, (Finkelstein, 2000), Limerick, Ireland.
- Glaser, N. & Derniame , J-C. (1998). *Software Agents: Process Models and User Profiles in Distributed Software Development*. 7<sup>th</sup> Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises (WETICE'98), Palo Alto, CA, USA.
- Grundy, J. C., Apperley, M. D., Hosking, J. G., & Mugridge, W. B. (1998). *A Decentralized Architecture for Software Process Modeling and Enactment*. IEEE Internet Computing, September/October.
- Høydalsvik, G.M. (1997). *Experiences in Software Process Modeling and Enactment*. PhD thesis, Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway.
- Humphrey, W. S. (1997). *Introduction to the Personal Software Process*. Addison-Wesley. Information Technology for European Advancement.
- Kahkonen, T. (2004). *Agile Methods for Large Organizations - Building Communities of Practice*. Agile Development Conference (ADC'04), Salt Lake City, Utah, USA.
- Kivisto, K. (1999). *Roles of Developers as Part of a Software Process Model*. Proc. of the 32<sup>nd</sup> Hawaii International Conference on System Sciences.
- Lacotte, J-P. (2004) *ITEA Report on Open Source Software*. Technical report, ITEA -
- Lehman , M. M. & Belady, L. A. (1985). *Program Evolution — Processes of Software Change*. Academic Press.
- McGrath, G. M. (1998). *Behavioural Issues in Software Engineering Process Modelling: A Multi-Paradigm Approach*. Hawaii International Conference on System Sciences (HICSS), January 1998.
- Naur, P. & Randell, B. (Eds.) (1969). *Software Engineering*. Proc. NATO Conference in Garmisch-Partenkirchen, 1968. NATO Science Committee, Scientific Affairs Division, NATO, Brussels.
- Nguyen, M. N. & Conradi, R. (1994). *Classification of Meta-processes and their Models*. Third International Conference on Software Process, Washington, USA.
- Nitto, E.D. & Fuggetta, A. (Eds.) (1998) *Process Technology: Special Issue*, volume 5 of Journal on Automated Software Engineering.
- OMG. (2002). *Software Process Engineering Metamodel Specification*. Formal/2002-11-14.
- Royce, W. W. (1987). *Managing the Development of Large Software Systems: Concept and Techniques*. In Proceedings of WesCon, 1970. Reprinted in Proc. Int'l Conf. Software Eng., IEEE Computer Society Press, 1987.

Scupin, R. (1997). *The KJ Method: A Technique for Analyzing Data Derived from Japanese Ethnology*. Human Organization, 56(2).

Sommerville, I. (1995). *Software Engineering*. Addison-Wesley. ISBN 0-2014-2765-6.

Straker, D. (1995). *A Toolbook for Quality Improvement and Problem Solving*. Prentice Hall International (UK) Limited.

Wang, A. I., Conradi, R. & Liu, C. (2000). *Integrating Workflow with Interacting Agents to support Cooperative Software Engineering*. 4<sup>th</sup> IASTED International Conference on Software Engineering and Applications (SEA'2000), Las Vegas, Nevada, USA.

Wang, A. I. (2000). *Using Software Agents to Support Evolution of Distributed Workflow Models*. International ICSC Symposium on Interactive and Collaborative Computing (ICC'2000) at International ICSC Congress on Intelligent Systems and Applications (ISA'2000), Wollongong, Australia.

Wang, A.I. (2001). *Using a Mobile, Agent-based Environment to Support Cooperative Software Processes*. PhD thesis, Norwegian University of Science and Technology. ISBN 82-7984-172-5.

Wang, A.I. (2002). *An Evaluation of a Cooperative Process Support Environment*. 6th IASTED International Conference on Software Engineering and Applications (SEA2002), Cambridge, MA, USA.

Wang, A.I., Larsen, J-O., Conradi, R. & Munch, B. (1998). *Improving Cooperation Support in the EPOS CM System*. 6th European Workshop on Software Process Technology (EWSPT'98), Weybridge (London), UK.

WfMC. (1999). *Workflow Management Coalition - Terminology & Glossary*. Technical report. The Workflow Management Coalition, February. Document Number WFMC-TC-1011, [http://www.wfmc.org/standards/docs/TC1011\\_term\\_glossary\\_v3.pdf](http://www.wfmc.org/standards/docs/TC1011_term_glossary_v3.pdf).

Yeomans, B. (1996). *Enhancing the World Wide Web*. Technical report, Computer Science Dept., University of Manchester. Supervisor: Prof. Brian Warboys.

## **ADDITIONAL READING**

The following is a list of references recommended for further reading.

Ambriola, V. , Conradi, R., Fuggetta, A. (1997). *Assessing process-centered software engineering environments*. ACM Transactions on Software Engineering and Methodology (TOSEM), v.6 n.3, p.283-328, July 1997

Andersson, C., Karlsson, L., Nedstam, J., Höst, M., & Nilsson, B. (2002). *Understanding Software Processes through System Dynamics Simulation: A Case Study*. Proceedings of the

9th IEEE Conference and Workshop on the Engineering of Computer-Based Systems, Lund, SWEDEN, 2002.

Barros, M.O., Werner, C.M.L. & Travassos, G.H. (2000). *Using Process Modeling and Dynamic Simulation to Support Software Process Quality Management*. XIV Simpósio Brasileiro de Engenharia de Software, Workshop de Qualidade de Software, João Pessoa, Brazil, October 2000.

Chatters, B. W., Lehman, M. M., Ramil, J. F., & Wernick, P. (1999). *Modelling A Software Evolution Process*. Software Process Modelling and Simulation Workshop, Silver Falls, Oregon, 28-30 June 99.

Collofello, J., Yang, Z., Merrill, D., Rus, I., & Tvedt, J. D. (1996). *Modeling Software Testing Processes*. The International Phoenix Conference on Computers and Communications (IPCCC'96), 1996.

Dean, D. L. , Lee, J. D. , Orwig, R. E. & Vogel, D. R. (1994). *Technological support for group process modelling*. Journal of Management Information Systems, v.11 n.3, p.43-63, December 1994

Deephouse, C., Mukhopadhyay, T., Goldenson, D. R., Kellner, M. I. (1995). *Software processes and project performance*. Journal of Management Information Systems, v.12 n.3, p.187-205, December 1995

Delen, D., Dalal, N. P., Benjamin, P. C. (2005). *Integrated modeling: the key to holistic understanding of the enterprise*. Communications of the ACM, v.48 n.4, p.107-112, April 2005

Estublier, J., Amieur, M., Dami, S. (1999). *Building a federation of process support systems*. ACM SIGSOFT Software Engineering Notes March 1999, Volume 24 Issue 2, Proceedings of the International Joint Conference on Work Activities Coordination and Collaboration WACC '99,

Gary, K., Lindquist, T., Koehnemann, H. & Sauer, L. (1997). *Automated process support for organizational and personal processes*. Proceedings of the international ACM SIGGROUP Conference on Supporting Group Work: The Integration Challenge, p.221-230, November 16-19, 1997, Phoenix, Arizona, United States

Glass, R. (1999). *The realities of software technology payoffs*. Communications of the ACM, v. 42, no. 2, p. 74-79. February 1999.

Gopal, A., Mukhopadhyay, T., Krishnan, M. S. (2002). *The role of software processes and communication in offshore software development*. Communications of the ACM, v.45 n.4, April 2002.

Gruhn, V. & Wolf, S. (1995). *Software process improvement by business process orientation*. Software Process--Improvement and Practice (Pilot Issue).

- Kahen, G., Lehman, M. M. & Ramil, J. F. (1999). *Empirical Studies of the Global Software Process - The Impact of Feedback*. Workshop on Empirical Studies of Software Maintenance (WESS-99), Sept. 3 - 4, 1999, Keble College, Oxford, UK.
- Keating, E., Oliva, R., Repenning, N., Rockart, S., & Sterman, J. (1999). *Overcoming the Improvement Paradox*. European Management Journal, 1999, 17(2), 120-134.
- Kellner, M. I., Madachy, R. J. & Raffo, D. M. (1999). *Software Process Modeling and Simulation: Why, What, How*. Journal of Systems and Software, Vol. 46, No. 2/3 (15 April 1999). p. 91-105.
- Kueng, P. & Kawalek, P. (1997). *Process models: A help or a burden?* Paper presented at the Association for Information Systems 1997 Americas Conference, August, Indianapolis.
- Lehman, M. M. (1997). *Feedback in the Software Process*. Software Engineering Association Easter Workshop, SEA'97, ICSTM, 14-15 April 97, pp. 43-49.
- Lehman, M. M. (1998). *Feedback, Evolution and Software Technology -The Human Dimension*. ICSE Workshop on Human Dimension in Successful Software Development, 20 Apr 98, Kyoto, Japan.
- Lin, C.Y. & Levary, R. R. (1989). *Computer-Aided Software Development Process Design*. IEEE Transactions on Software Engineering 15(9). pp. 1025-1037 .
- Madachy, R. & Tarbet, D. (2000). *Case studies in software process modeling with system dynamics*. Software Process: Improvement and Practice Volume 5, Issue 2-3, 2000. Pages: 133-146.
- Martin, R. H. & Raffo, D. M. (2000). *A Model of the Software Development Process Using Both Continuous and Discrete Models*. Software Process: Improvement and Practice Volume 5, Issue 2-3, 2000. Pages: 147-157.
- McGrath, G. M.. (1996). *Representing Organisation and Management Theories in Software Engineering Process Modelling*. Proceedings of the IASTED Conference, August 19-21, 1996, Honolulu, Hawaii.
- McGrath, G. M., Campbell, B., More, E., & Offen, R. J. (1999). *Intra-Organisational Collaboration in a Complex, Rapidly-Changing Information services company: A Field Study*. Proceedings ISDSS'99, Melbourne, Pacific Mirror Image, Melbourne, Australia, 1999, ISBN 0732620740
- McGrath, G. M. (1997). *A Process Modelling Framework: Capturing Key Aspects of Organisational Behaviour*. Australian Software Engineering Conference (ASWEC '97) September 28 - October 2, 1997, Sydney, AUSTRALIA .
- Muehlen, M. Z. (2004). *Organizational Management in Workflow Applications – Issues and Perspectives*. Information Technology and Management, v.5 n.3-4, p.271-291, July-October 2004

Mishali, O. & Katz, S., *Using aspects to support the software process: XP over Eclipse*. Proceedings of the 5th International Conference on Aspect-Oriented Software Development, March 20-24, 2006, Bonn, Germany

Oliveira, T. C., Alencar, P. S. C., Filho, I. M., de Lucena, C. J. P. & Cowan, D. D. (2004). *Software Process Representation and Analysis for Framework Instantiation*. IEEE Transactions on Software Engineering, v.30 n.3, p.145-159, March 2004

Paech, B., Dorr, J. & Koehler, M. (2005). *Improving Requirements Engineering Communication in Multiproject Environments*. IEEE Software, v.22 n.1, p.40-47, January 2005

Raffo, D. M., Harrison, W. & Vandeville, J. (2000). *Coordinating Models and Metrics to Manage Software Projects*. Software Process: Improvement and Practice Volume 5, Issue 2-3, 2000. p. 159-168.

Richardson, G. P. & Andersen, D. (1995). *Teamwork in group model building*. System Dynamics Review, 11(2), 113-137.

Russell, N., van der Aalst, W. M. P., ter Hofstede, A. H. M. & Wohed, P. (2006). *On the suitability of UML 2.0 activity diagrams for business process modelling*. Proceedings of the 3rd Asia-Pacific Conference on Conceptual Modelling, p.95-104, January 16-19, 2006, Hobart, Australia

Sawyer, S. & Guinan, P. J. (1998). *Software development: Processes and performance*. IBM Systems Journal, Vol. 37, No. 4, 1998.

Scacchi, W. (1999). *Understanding Software Process Redesign using Modeling, Analysis and Simulation*. ProSim'99 Workshop on Software Process Simulation and Modeling, Silver Springs, OR 27-29 June 1999.

Sharp, A. & McDermott, P. (2001). *Workflow Modeling: Tools for Process Improvement and Application Development*, Artech House, Inc., Norwood, MA, 2001

Stelzer, D. & Mellis, W. (1998). *Success factors of organizational change in software process improvement*. Software Process: Improvement and Practice, 4, p. 227-250.

Verlage, M. (1996). *About views for modeling software processes in a role-specific manner*, Joint proceedings of the Second International Software Architecture Workshop (ISAW-2) and International Workshop on Multiple Perspectives in Software Development (Viewpoints '96) on SIGSOFT '96 workshops, p.280-284, October 16-18, 1996, San Francisco, California, United States

Wang, Y. & Bryant, A. (2002). *Process-Based Software Engineering: Building the Infrastructures*. Annals of Software Engineering. Volume 14 Issue 1-4, December 2002, J. C. Baltzer AG, Science Publishers