Towards a Tool Support for a Living Software Development Process [1] [2]

Michael Gnatz, Frank Marschall, Gerhard Popp,
Andreas Rausch and Wolfgang Schwerin
Technische Universität München, Arcisstr. 21, D-80290 München, Germany
{gnatzm|marschal|popp|rausch|schwerin}@in.tum.de

Abstract

Change and evolution of business and technology imply change and evolution of development processes. Besides that for a certain enterprise or a project we will usually integrate elements from a variety of existing process models, comprising generic standards as well as specific development methods. In this paper we propose a process model framework which is modularly structured on the basis of the concept of Process Patterns. This framework allows us to describe development processes in a way such that change, evolution, and integration of processes are facilitated. Founded on our framework we sketch the idea of a Living Software Development Process. An important step in this direction is a proper knowledge management tool support. In this paper we discuss and propose the approaches and requirements to fulfill the users specific needs of a sophisticated tool support for a Living Software Development Process.

1. Introduction

Nowadays, many different software development process models exist. These models range from generic ones, like the waterfall model [21] or the spiral model [6], to detailed models defining not only major activities and their order of execution but also proposing specific notations and techniques of application. Examples of the latter kind are the Objectory Process [13], the Unified Software Development Process [15], the Catalysis Approach [11], the V-Modell 97 [12], or eXtrem Programming [4] – just to name some of them.

All these process models have their individual assets and drawbacks. Hence, one would wish to take all the different assets and benefits of the various process models as a basic construction kit for an integrated development process tailored to the specific needs of the individual team, project, company, and customer.

To assemble a specific development process from existing models we have to identify and characterize explicitly the building blocks and their relations of a process model in general. Therefore we need a set of basic notions and definitions common for all process models – a so called process model framework. This framework must allow us to integrate the various existing process models. The process model framework can serve as a common basis for the definition of a development process that incorporates the assets and benefits of the different existing process models and that can be flexibly adapted to different kinds of project requirements and situations.

Once you have defined your standardized development process in terms of the framework, you still have to adapt this development process to different projects and project situations. This is often referred to as static tailoring. But, our business is changing almost every day: the requirements of our customers change, new technology has to be adopted, and finally the way we work together evolves. To be successful in a changing environment we not only need static adaptation but also a more flexible way of adaptation - the dynamic adaptation, i.e. the openness and flexibility to enhance the process model itself while the process is enacted within a project.

Tom DeMarco even mentioned about the nature of process models and methodologies in [10]: „It doesn't reside in a fat book, but rather inside the heads of people carrying out the work." Thus, our process model framework must additionally offer the ability to incorporate the process knowledge of the whole company. It must provide a platform for a learning organization recording the evolution steps of a Living Software Development Process.

In this paper we introduce our process model framework that is discussed more in detail in [30]. However, a framework like this can only add value to an organization if it is applied reasonable and supported with appropriate tools in practice. Thus the main objective of this paper is to demonstrate how our approach of a Living Software Development Process can be adopted in real world organizations.

---

[1] This work originates form the research project ZEN – Center for Technology, Methodology and Management of Software & Systems Development – a part of Bayerischer Forschungsverbund Software-Engineering (FORSOFT), supported by the Bayerische Forschungsstiftung.

Since we strongly believe that the application of such a flexible and dynamic approach must be accompanied by a sophisticated tool support we will focus on this tool support to demonstrate how our framework can be used successfully.

The desired tool support must address dynamic as well as static adaptation of the development process models. Moreover the tool has to support the project team during the whole lifecycle of the development project.

In the following section we give an overview of the different levels of development processes and models, namely the project, method and metamodel level. We present the requirements of an evolutionary process model based on these views. In the next section, Section 3, we define our process model framework. In Section 4 we present scenarios for the application of our approach and extract the requirements of a sophisticated tool support for the presented concepts, including definition and improvements of process models as well as support for the project level. A conclusion is given at the end of the paper in Section 5.

2. The Living Software Development Process

Various people and groups get into touch with process models and metamodels. In the following two sections, we present the two main views on process models – the project view and the method view. We will discuss their specific way of interaction with the Living Software Development Process we are going to propose in this work. Thus, we can show the needs and benefits of the two different user groups mentioned above. Finally, we present the different levels of a software development process and show how tool support is related to these levels.

2.1. The Project View

Companies, which are on Capability Maturity Model (CMM) level 3 or higher, have a standardized process model [20]. This standard process model provides guidelines and support for an organization's projects in general. A tool is needed to document and offer the standard process models to project managers and project team members. While performing their daily tasks they are creating documents that follow predefined templates. The tool should offer these templates and the applications to edit the templates.

Managing a concrete project implies the selection of a suitable process from a set of existing, possibly standardized alternatives as shown in Figure 1. This process can flexibly be build by iteratively choosing Process Patterns as proposed in this paper. Decision support is needed to find an appropriate process for a concrete project. Once a process is chosen, it has to be tailored dynamically according to the project's situation, i.e. more fine-grained Process Patterns have to be chosen as the project continues. Again a proper tool support is needed to tailor the development process to the specific needs of the project. The tailored process represents the guidelines, which are intended to provide guidance for project team members.

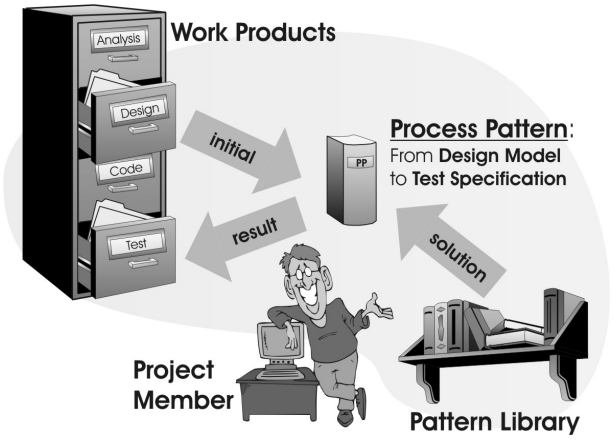

Figure 1: Project view on the Living Software Development Process

In terms of our process model framework, given in Section 3, the tailored process defines which Process Patterns are to be applied in a concrete project, which modeling concepts and notations are to be used and also which work products are to be produced.

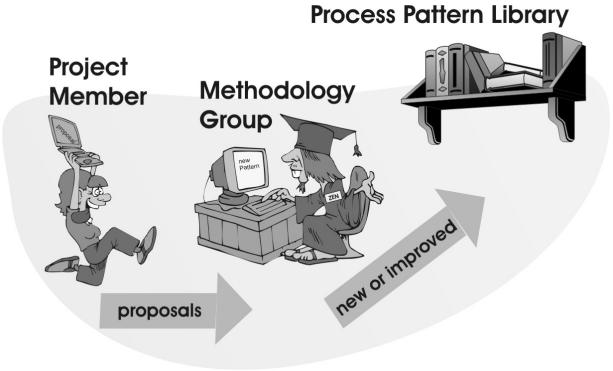

Figure 2: Method view on the Living Software Development Process

2.2. The Method View

Process improvement, as required on CMM level 5 [20] for example, means the evolution of process models. The formulation of process models on the basis of a well-defined ontology facilitates comprehension and hence

changes of development processes. Elements of the process model framework are supposed to play the role of such an ontology defining terms like “Activity”, “Process Pattern”, “Work Product” and their interrelations.

An ontology for development processes provides both, the developers and the methodology group, with a common vocabulary. On the one hand a methodology group can use such an ontology for the definition of standardized processes. On the other hand developers can use this vocabulary for the description of proposals for changes or additional process elements, which reflect their experience made with formerly applied processes. On the basis of these proposals redefinitions by the methodology group can be done.

Figure 2 shows the method view on the Living Software Development Process. The Process Pattern library which is depicted by a bookshelf has to be supported by a proper tool as proposed in this paper.

2.3. Process Models and Metamodels

According to [28, 29] we can divide the software development process into the production process, that is the process in charge of developing and maintaining the product to be delivered, and the meta process, that is the process responsible for maintaining and evolving the whole software process. Using this terminology, we may see the project view as the production process, and the methodology view as the meta process.

Figure 3 combines these concepts into an overall model for software development processes where the aforementioned views can be mapped on.

The Instance Level in Figure 3 captures those elements that belong to a certain process in a certain project, such as analysis documents or a certain Process Pattern applied in a concrete project, for example. This level corresponds to the project view already mentioned.

The Model Level describes a certain software development process. This process definition might contain a description of an analysis document, guidelines of how to develop software according to a waterfall process model or guidelines of how to organize and hold a workshop with customers to elicit requirements. This level offers guidelines for project managers as well as team members. A specific process model, as defined in [28], expressed in a suitable process modeling language, would be an element of the Model Level.

The Metamodel Level provides the basic framework to establish a living process model. It offers clear definitions for terms like „Work Product”, “Process Pattern” or „Activity”. The Metamodel Level represents the common conceptual base for a company’s methodology group to improve and evolve the underlying standard software development process model. It is on this level where the concepts of process modeling languages, such as EPOS SPELL and SOCCA (cf. [9, 28]) are defined.

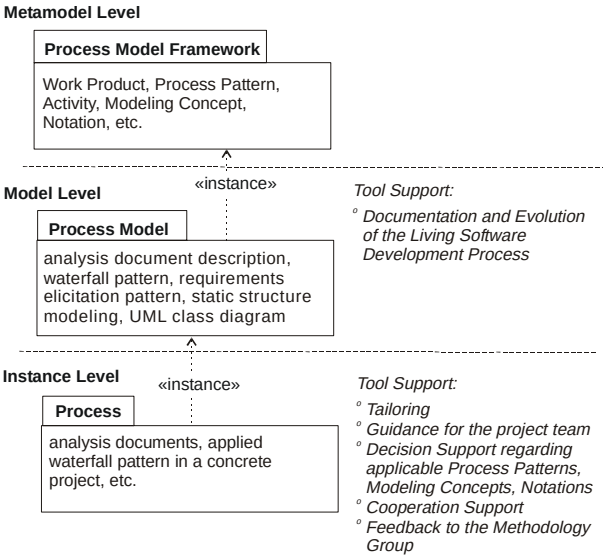

Figure 3: Overall Model of the Living Software Development Process

On each of these levels specific features of a sophisticated and universal tool support can be identified. On the Model Level, a tool has to deal with documentation and evolution of the proposed process model. Supporting this level with a suitable tool is the prerequisite for supporting the Instance Level of a living development process. On the Instance level some among many desired features are tailoring of the generic model to a project-specific one, guidance of the project team, decision support regarding applicable Process Patterns, modeling concepts and notations, as well as cooperation support. Next to supporting the project an important requirement is to provide a feedback loop to the methodology team to ensure the evolution of the process on the Model level. These high-level requirements of tool support are further elaborated in Section 4.

3. The Process Model Framework

In the previous section we have shown how developers and the methodology group may interact for elaborating and improving a standard software development process establishing a Living Software Development Process. Our basic ontology is defined in the process model framework, which is on the Metamodel Level in Figure 3.

The framework must provide the ability to define and maintain a process model, which integrates elements of all the various existing process models, like for instance the Rational Unified Process [17] or the V-Modell 97 [12].

Thus, the framework must enable the methodology group to state clearly the correlations between the elements of the different process models in terms of a common meta model, to allow their seamless integration. Additionally, the framework must support static as well as dynamic adaptation of the process model with respect to the evolution and learning of a living organization.

The new, upcoming concept of Process Patterns seems to be an approach which basically follows our ideas and which may fulfill our requirements. Process patterns are a very general approach allowing us to integrate existing process models without having to develop a new model from scratch [2, 3, 7, 8]. For example in [1] we have already shown the integration of the V-Modell in the Process Pattern approach.

The basic idea of the concept of Process Patterns is to describe and document process knowledge in a structured, well defined, and modular way. Moreover patterns provide information helping us in finding and selecting alternative development steps, similar to strategies and selection guidelines in [22]. Conform with most authors, patterns in our approach consist mainly of an initial context, a result context, a problem description and a solution. The initial context is an overall situation giving rise to a certain recurring problem that may be solved by a general and proven solution. The solution leads to the result context [5].

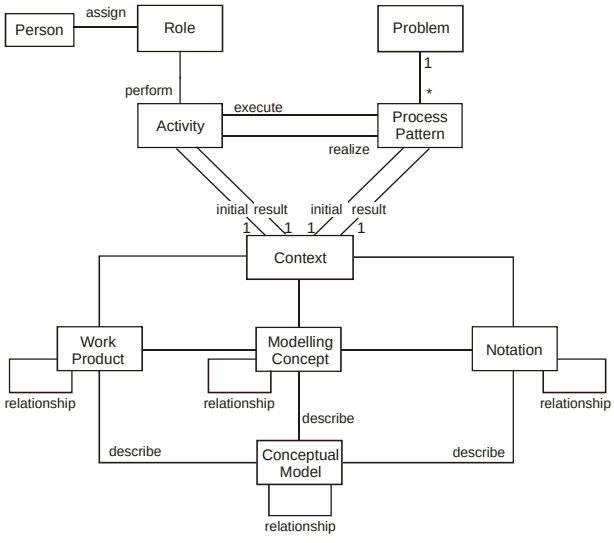

Figure 4: Process Model Framework

Figure 4 illustrates the basic concepts of the proposed process model framework. It develops further the Process Pattern approach from [7, 8], and integrates it with an enhanced variant of the widely accepted process model framework given in [9]. The framework is based on a clear separation of concerns between the overall result structure, the consistency criteria, and the Process Patterns themselves.

A Process Pattern defines a general solution to a certain recurring problem. The problem mentions a concrete situation that may arise during the system development. It mentions internal and external forces, namely influences of customers, competitors, component vendors, time and money constraints and requirements. A Process Pattern suggests an execution of a possibly temporally ordered set of activities. Activities themselves may be carried out following the guidelines of other, subordinated Process Patterns realizing the activity in question. Therefore Process Patterns and activities in our framework may be structured hierarchically, but iterative activities like the application of the spiral model [6] may also result in more complex structures that contain loops.

Process Patterns in our framework represent strategies to solve certain problems. Activities represent development steps and are executed by Process Patterns. An activity does only describe what is to be done but not how it is to be done. In contrast to that a Process Pattern provides a solution for realizing an activity. Hence generally one activity might be realized by different Process Patterns. Activities are performed by definite roles. In turn roles are assigned to corresponding persons.

Each Process Pattern as well as each activity needs an initial context to produce a result context. The initial context describes the required project situation to perform an activity or pattern, respectively. The result context describes the situation we reach when performing an activity or pattern, respectively. The context captures the internal state of the development project and can be characterized by constraints over the set of work products. Simple constraints are that certain work products have to exist.

A process model assigns certain Process Patterns, as for instance the pattern “Planning the Project”, to certain work products, as for example the “Project Schedule”. These work products are described by means of modeling concepts, as for instance “Time Flow Modeling”. The modeling concepts are represented by certain notations, such as “UML Sequence Diagrams”.

The initial and result context of a Process Pattern may not only require the existence of certain work products, but also that certain modeling concepts and notations are to be applied for these work products. This is important when a pattern proposes the application of notation specific techniques. For instance in [18] methodical guidelines for the refinement of specifications are introduced. These refinement techniques require the modeling concept “Interaction Modeling” based on the notation “Message Sequence Charts”.

For the executes and realizes relationships in Figure 4 we require certain relationships between the contexts of related Process Patterns and activities. The work products in the result context of a Process Pattern have to be a su-

perset of the result context of each realized activity. The initial context of a Process Pattern yet has to be a subset of the initial context of each realized activity. With these consistency criteria we cover the intuition that a realizing pattern does require at most the input of the realized activity, to produce at least those results "promised" by the activity.

Consistency is also required for the contexts regarding the executes relationship. The union of the result contexts of the executed activities form the result context of the executing Process Pattern. The initial contexts of the activities have to be subsets of the initial contexts of the Process Pattern they are executed by. Thus inter-mediate results produced in the workflow of the executed activities need not necessarily be part of the initial context of the executing activity. Preserving consistency is an important feature of our tool support proposed in Section 4.

The precise definition of the meaning of, and context conditions between work products can be achieved by the use of a so-called conceptual model. Work products that are based on sound description techniques have not only a well-defined notation, but also a possibly even formal semantics in form of a mapping from the set of work products into the set of systems (cf. [16, 23, 25]). A conceptual model characterizes, for instance, the set of all systems that might ever exist. This integrated semantics provides the basis for the specification of a semantic preserving translation from specification work products to program code. This can serve as a basis for correct and comprehensive code generation.

The circular relationship associations assigned to various elements in Figure 4, such as work product and conceptual model, cover the general idea of structuring these elements, for example hierarchically.

4. Tool Support for a Living Process

In contrast to "traditional" methods of software development the pattern-based Living Software Development Process cannot be documented very well within a static medium such as a book or static HTML-Sites in a company's intranet. The Living Software Development Process represents the methodological knowledge base of a company or organization. However, it is typical for knowledge that its amount, structure and content change over time while the organization gains more experience and professionals as well as science provide new solutions and also new problems. Thus we strongly believe that the application and the definition of a Living Software Development Process needs a much more flexible medium that allows not only its documentation but also its maintenance in real-time. Therefore we are currently developing the web-based "Living Software Development Process Support Application" (LiSA), which allows the definition, maintenance and documentation of Process Patterns in terms of our process model framework (cf. Section 3).

In this section we present a concrete scenario to demonstrate the application of our Process Pattern approach. We will show how methodologists and projects can make use of appropriate tools like LiSA to introduce and apply a Living Software Development Process. Finally, we identify the basic concepts and requirements of such a tool and show how to support projects applying the Living Software Development Process.

4.1. Tool support for the Methodology Team

In our scenario we will start with the methodology group of a larger software development company called "SuperSoft". This group is responsible to define and document the development process for the different projects at SuperSoft.

SuperSoft's methodology team would like to be able to combine different existing development methods with their own experiences to provide projects with a flexible set of Process Patterns that can be combined to an ideally adapted process. Therefore SuperSoft's methodology team decides to apply the tool LiSA to establish the Living Software Development Process we introduced in the previous sections. LiSA allows defining a complete pattern-based process at the Model Level (cf. Figure 3) and thus integrating various existing methods with own approaches.

First the methodology team has to define the set of work product descriptions and templates according to those a project develops software. The work product structure serves as a common basis for the integration of development process knowledge. Most of these work product definitions may already exist in some way, since the projects use a lot of existing templates like templates for test reports, use case descriptions, and so on. Thus, the methodology team has to build up a description of a whole work product filing cabinet by hierarchically arranging the work product structure in the tool. For every work product definition there is a short description and an explanation of the work products purpose. Furthermore, there are links that lead to available templates and best practice samples for every work product.

After the creation of a complete filing cabinet the methodology team uses the tool to define typical activities that may be performed during software development. Typically, they will choose activities that may be performed in different ways. Hence, the realization of these may be described by different Process Patterns. For every activity they give a short explanation of the development issue and mark a set of work products in their filing cabinet as necessary input and another set as guaranteed output. For example for the new activity "Testing" the work

products “Specification” and “Code” may serve as input while the output are work products of the type “Test Report”. Therefore, the tool allows the definition of input and output contexts, just by selecting the appropriate work product descriptions from the tools filing cabinet tree.

Now SuperSoft has already a consistent description of what has to be produced and which artefacts are needed to produce other artefacts. In the next step SuperSoft’s methodology team starts to document the companies existing processes.

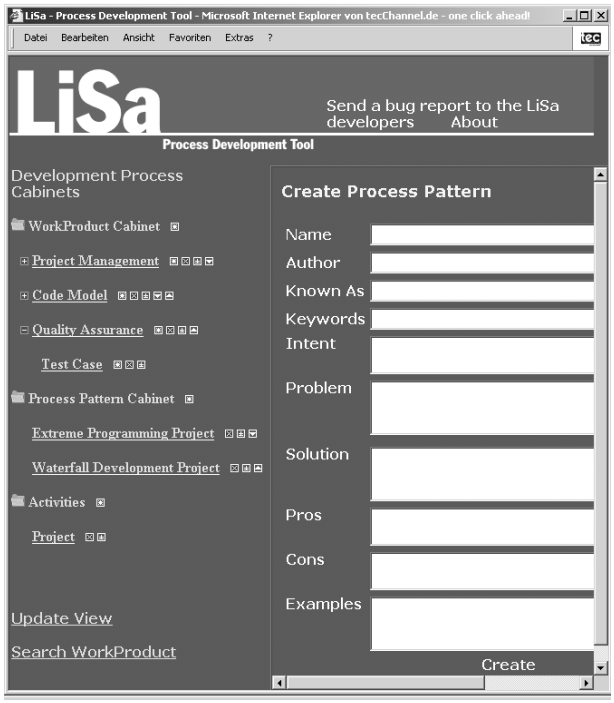

Figure 5: Defining a new Process Pattern within LiSA

Figure 5 shows an example dialog from our process development tool LiSA. The dialog for editing a Process Pattern allows filling the description fields of the pattern template which make up a pattern description, as for example intent, problem and solution. In the left frame of the dialog in Figure 5 some of the other editable entities of our process model framework are shown in a tree structure.

When creating a new Process Pattern the user can define an input and an output context like when creating a new activity. After that LiSA offers a set of already defined activities that the new pattern may realize according to the consistency criteria explained in Section 3. Human interaction is then required to determine the intended, i.e. semantically realized activities among those suggested by the tool.

The pattern’s description also comprises a description of all the executed activities and how their different inputs and outputs relate. Therefore the user can enter plain text. Whenever he wants to refer to an executed activity he can select this activity from the list of activities and LiSA will record a new “executes” relationship between pattern and activity.

Based on the already given work product filing cabinet and the activities SuperSoft’s methodology team defines a set of Process Patterns that may originate from various established methods and/or from own experience different SuperSoft projects have gained over time. During the process of writing new Process Patterns it may be necessary to add new activities or work product descriptions to integrate all kinds of methodological approaches. So in the end SuperSoft has created its own process model for a Living Software Development Process.

Probably it may be necessary to define several of these pattern-based development processes to support very different kinds of projects, such as the development of web-based business applications and embedded controller software. These process definitions may originate from a common set of core work products and patterns, which help to integrate the company’s processes.

A Living Software Development Process is alive, not only because it offers flexibility to those who apply it, but also because it allows to be extended and modified whenever necessary or desired. It has to be ensured, that the consistency of the complete process definition is not violated by modifications of the process. For example, a change in the input context of an activity may result in the fact that several patterns don’t realize this activity syntactically any more while some others could do so now. So, whenever a member of SuperSoft’s methodology team does perform such a modification, she will be informed about the impacts and may chose additional steps to keep the process definition sound and easy to apply.

Of course, the tool also allows browsing the resulting set of work product descriptions, activity descriptions, and Process Patterns. Thus LiSA can also be used by the projects that apply the company’s new living process as a comfortable process documentation.

Project team members can submit proposals via LiSA’s web interface and the people of the methodology group enter a consolidated and sound description of the proposed or enhanced pattern including the problem domain, when the pattern’s application is advisable and its purpose. Involving project members in the definition and later in the improvement of Process Patterns helps to ensure that the defined process is lived within the company.

As we have seen, a tool like LiSA must meet a set of requirements to allow the definition of a company’s own Living Software Development Process. This includes

- the definition and maintenance of
  - a work product filing cabinet,
  - development activities,
  - Process Patterns with consistent realizes and executes associations,

- and the ability to browse and search within the process definition to use it as an integrated guide book.

The LiSA tool serves as documentation of the process and therefore can be used as a guide by projects. However, our vision of a more sophisticated tool support for projects is introduced in the next section.

4.2. Tool support for Projects

There are already web-based documentations of methods like the Rational Unified Process [17] or Catalysis that offer a set of development activities and work product descriptions. However, all these approaches are stuck with a given methodology and usually they do not deal with instances of work products and processes in software development. Hence in this paper we suggest our vision of an enhanced tool support for projects that also deals with concrete instances of work products and activities – not just their definition. The proposed tool support for projects is based on our generic process model framework (cf. Figure 4). Thus it does not come along with the restrictions of currently available tools.

While developing the tool LiSA to support the method view our group is already planning to extend this tool to support projects in the application of a pattern-based development process. To illustrate these considerations we extend our scenario and show how development projects can profit from such an approach and how a tool can support them. In the extended scenario we take a closer look at the project "Banking Account Management System" (BAMS) at SuperSoft.

When the project manager of BAMS initially begins to use the extended tool "LiSA for Projects" (LiSA-Pro) he selects an appropriate software development process model defined by SuperSoft's methodology team. In this case he might chose SuperSoft's standard process model for information systems development.

Next the project manager tailors the filing cabinet, i.e. he starts to drop out certain work products his specific project does not need. For example, if BAMS is build from scratch without the need to fit in an existing environment, he might drop out all the work products concerning the "As-Is Analysis". Consequently the tool will not show him any development activities or Process Patterns that rely on these work products in the further course of the project.

After tailoring the selected work product filing cabinet the project manager instantiates it. Therefore the tool generates a complete directory structure of all work products in the project's home directory and fills it with the available work product templates. For every work product description initially one instance exists. If a project member wants to create an additional work product instance he can do so at any time. The tool will add the new instance to the tool's work product instances tree and create a new empty template for the instance. For example the work product description "test case" naturally gets instantiated several times during the BAMS project.

For each work product instance the status "empty", "under work", "released" is tracked by the tool. Initially all work product instances are "empty". The status of these instances has to be coupled with a sophisticated version control system. This allows the project members to use their version control system to reconstruct old versions of the work product instances including their state.

Now, after the project's structure has been created for BAMS the project team can start filling it with work products that are initially present, such as contracts with customers, a vision statement or even existing code from a predecessor project can be reused. The new work products replace the empty templates and their state is set to "released", which indicates that the work product may be used as input for further activities.

At this point all initial configuration work is done and the team can start "productive" work. As the project manager looks at the tool he gets offered a list of applicable development activities he can perform with the given set of "released" work products.

As a first step the project manager chooses a very coarse grained activity called "Project". The tool offers him a set of applicable Process Patterns that realize this activity. These might be for example the "Extreme Programming-Project Pattern" or the "Waterfall Development-Project Pattern" that are based on well known existing methods [4, 21]. The manager reads the problem descriptions, Pros and Cons of these patterns and then decides which description fits best for the situation of the BAMS project. For example there might be a pattern that is very recommendable because it fits to project situations of high time pressure.

Whenever a user decides to perform a certain Process Pattern he chooses the pattern in the tool. In an additional dialog he can choose the work product instances he wants to lock, e.g. if someone decides to perform a testing pattern he has to select at least one module of code he wants to test, but not all work products of the type "code". All the selected work products are marked "under work" and thus locked within the revision control system. After the pattern has been applied all the work products of the initial and the result context are set to "released", until no other user has set some of them "under work" again.

From now on most of the time there will be an active Process Pattern that guides the project team through its work. The tool indicates all these active patterns (or processes) and lists the activities that still have to be performed to finish each process. In choosing the appropriate patterns to realize these activities the project can flexibly and dynamically react to external forces such as changing

requirements, the loss of team members or budget cuts at any time throughout the project.

The focus of the LiSA tool is on the guidance of the project team members in their daily work – the enforcement of a prescriptive process is not addressed. Next to supporting projects the focus of the tool is to narrow the sometimes existing gap between methodology departments and daily project work.

Therefore after offering a Process Pattern as a guide to a project team member the tool automatically provides a feedback loop to the methodology team. Thus the tool stimulates the recording of experiences, pros and cons of patterns, suggestions for improving existing patterns or for new patterns based on experiences gained within a project. Therefore the tool is also intended to support the methodology team with process elicitation and to ensure that the process is evolving and thus living.

5. Conclusion

In this paper we have introduced a framework for a Living Software Development Process that is based on the idea of Process Patterns. We discussed the project view and the method view on processes. On top of these views we established our process model framework that allows us to integrate and evolve existing and new processes models.

We have shown that a pattern-based approach in software engineering is a great benefit for both, methodology group and project teams. With an appropriate tool support as presented in this paper the application of this approach appears to be very promising and can be applied for both of the two views on processes.

At the moment we are developing the J2EE-based web application LiSA that helps to define and maintain an organization's software development process. Thus LiSA may serve an organization as an enterprise-wide knowledge management platform that provides all the information about a companies development processes. The concept of Process Patterns allows the flexible adoption of the process at the method level, while it enables the software developers to choose the process that fits best for their individual project situation at any time.

For most projects the outlined tool support will be a great added value. However there are some features someone might still miss. For example a sophisticated workflow and cooperation support or tighter integration with other development tools are desirable. Furthermore project managers might want additional help for choosing the right Process Pattern in a certain situation. Therefore we are currently planning the integration of the tool ProChoiceII[3] that helps to choose an appropriate Process Pattern for cost estimation depending on certain project parameters.

6. References

[1] Dirk Ansorge, Klaus Bergner, Bernd Deifel, N. Hawlitzky, C. Maier, Barbara Paech, Andreas Rausch, Marc Sihling, Veronika Thurner, Sascha Vogel. Managing Componentware Development - Software Reuse and the V-Modell Process. In Lecture Notes in Computer Science 1626, Advanced Information Systems Engineering, Page 134-148, Editors Matthias Jarke, Andreas Oberweis. Springer Verlag. 1999.

[2] Scott W. Ambler. Process Patterns: Building Large-Scale Systems Using Object Technology. Cambridge University Press. 1998.

[3] Scott W. Ambler. More Process Patterns: Delivering Large-Scale Systems Using Object Technology. Cambridge University Press. 1999.

[4] Kent Beck. Extreme Programming Explained: Embrace Change. Addison-Wesley. 1999.

[5] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal. Pattern-Oriented Software Architecture, A System of Patterns. John Wiley & Sons. 1996.

[6] Barry Boehm. A Spiral Model of Software Development and Enhancement. ACM Sigsoft Software Engineering Notes, Vol. 11, No. 4. 1986.

[7] Klaus Bergner, Andreas Rausch, Marc Sihling, Alexander Vilbig. A Componentware Development Methodology based on Process Patterns. Proceedings of the 5th Annual Conference on the Pattern Languages of Programs. 1998.

[8] Klaus Bergner, Andreas Rausch, Marc Sihling, Alexander Vilbig. A Componentware Methodology based on Process Patterns. Technical Report TUM-I9823, Technische Universität München. 1998.

[9] J.-C. Derniame, B. Ali Kaba, D. Wastell (eds.): Software Process, Principles, Methodology, and Technology. Lecture Notes in Computer Science 1500, Springer, 1999.

[10] Tom DeMarco, Timothy Lister. Peopleware, Productive Projects and Teams, Second Edition Featuring Eight All-New Chapters. Dorset House Publishing Corporation. 1999.

[11] Desmond Francis D'Souza, Alan Cameron Wills. Objects, Components, and Frameworks With Uml: The Catalysis Approach. Addison Wesley Publishing Company. 1998.

[12] Wolfgang Dröschel, Manuela Wiemers. Das V-Modell 97. Oldenbourg. 1999.

[13] Ivar Jacobson. Object-Oriented Software Engineering: A Use Case Driven Approach. Addison Wesley Publishing Company. 1992.

[14] Ivar Jacobson. Component-Based Development Using UML. Invited Talk at SE:E&P'98, Dunedin, Newzealand. 1998.

[3] The tool ProChoiceII is also developed within the research project FORSOFT.

[15] Ivar Jacobson, Grady Booch, James Rumbaugh. Unified Software Development Process. Addison Wesley Publishing Company. 1999.

[16] C. Klein, B. Rumpe, M. Broy: A stream-based mathematical model for distributed information processing systems - SysLab system model. In Proceedings of the first International Workshop on Formal Methods for Open Object-based Distributed Systems, Chapmann & Hall, 1996.

[17] Philippe Kruchten. The Rational Unified Process, An Introduction, Second Edition. Addison Wesley Longman Inc. 2000.

[18] Ingolf Krüger. Distributed System Design with Message Sequence Charts. Dissertation, Technische Universität München. 2000.

[19] Object Management Group (OMG). Meta Object Facility (MOF) Specification. http://www.omg.org, document number: 99-06-05.pdf. 1999.

[20] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber. Capability Maturity Model for Software, Version 1.1. Software Engineering Institute, CMU/SEI-93-TR-24, DTIC Number ADA263403. 1993.

[21] Winston W. Royce. Managing the Development of Large Software Systems: Concepts and Techniques. In WESCON Technical Papers, Western Electronic Show and Convention, Los Angeles, Aug. 25-28, number 14. 1970. Reprinted in Proceedings of the Ninth International Conference on Software Engineering, Pittsburgh, PA, USA, ACM Press, 1989, pp. 328-338.

[22] C. Rolland, N. Prakash. A. Benjamen: A multi-Model View of Process Modelling. Requirements Engineering Journal, to appear.

[23] Bernhard Rumpe: Formale Methodik des Entwurfs verteilter objektorientierter Systeme. Herbert Utz Verlag Wissenschaft, 1996.

[24] A.-W. Scheer: ARIS, Modellierungsmethoden, Metamodelle, Anwendungen. Springer Verlag, 1998.

[25] B. Schätz, F. Huber: Integrating Formal Description Techniques. In: FM'99 - Formal Methods, Proceedings of the World Congress on Formal Methods in the Development of Computing Systems, Volume II. J. M. Wing, J. Woodcock, J. Davies (eds.), Springer Verlag, 1999.

[26] OMG: Unified Modeling Language Specification, Version 1.3 alpha R5, March 1999, http://www.omg.org/.

[27] Workflow Management Coalition: Terminology & Glossary. Document Number WFMC-TC-1011, Status 3, www.wfmc.org, February 1999.

[28] A. Finkelstein, J. Kramer, B. Nuseibeh: Software Process Modelling and Technology. Research Studies Press Ltd, JohnWiley & Sons Inc, Taunton, England, 1994.

[29] R. Conradi, C. Fernström, A.Fuggetta, R. Snowdon: Towards a Reference Framework for Process Concepts. In Lecture Notes in Computer Science 635, Software Process Technology. Proceedings of the second European Workshop EWSPT'92, Trondheim, Norway, September 1992, pp. 3-20, J.C. Derniame (Ed.), Springer Verlag, 1992.

[30] M. Gnatz, F. Marschall, G. Popp, A. Rausch, W. Schwerin: Towards a Living Software Development Process based on Proscess Patterns, Proceedings of the Eight European Workshop on Software Process Technology 2001, Lecture Notes in Computer Science 2077, V. Ambriola (editor), pp. 182-202, Springer, 2001