Modular Process Patterns supporting an Evolutionary Software Development Process[1]

Michael Gnatz[1], Frank Marschall[1], Gerhard Popp[1],
Andreas Rausch[1] and Wolfgang Schwerin[1]

[1] Technische Universität München, Institut für Informatik,
Lehrstuhl Professor Dr. Manfred Broy,
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 and/or 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 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 development process. An example illustrates our approach.

1 Introduction

Nowadays, many different 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. Jacobson, for example, talks about the unified process as a “strawman” serving only for explanatory purposes and probably never applied exactly as proposed [14].

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

[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*.

model in general. Therefore we need a set of basic notions and definitions common for all process models – the Process Framework. This Process Framework must allow us to integrate the various existing process models. The Process 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 Process 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.

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 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.

Therefore the Process Framework must be open for the integration of new process elements, for the extension and adaptation to new technologies and fields of application. Besides static adaptation – support of different kinds of projects and project situations – there is a need of dynamic adaptation.

In this paper we propose a Process Framework that is sufficiently powerful to fulfill these requirements. First, in Section 2 we give an overview over different modeling levels of development processes. We present the requirements on and user views of an evolutionary process model. In the next section, Section 3, we define our Process Framework. In Section 4 we present the concept of process patterns, providing guidelines about the organization of the actual development process. A conclusion is given at the end of the paper in Section 5.

2 Basic Concepts of the Living Software Development Process

Various people and groups get into touch with process models and metamodels. In the next section, we discuss the different levels of a software development process. Then, 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.

2.1 Process Models and Metamodels

Developing and maintaining software has become one of the most challenging tasks a company can do. Following some kind of process model, such as the Rational Unified Process [17] or the V-Modell 97 [12], promises to provide guidance facilitating the development task. Usually there are different people involved with different views on the development process itself.

Developers and project leaders are concerned about the development process of their individual projects. They concentrate on concrete tasks and results of their actual project, like the development of an analysis model of the system under consideration. Accordingly 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 products to be delivered, and the *meta process*, that is the process in charge of maintaining and evolving the whole software process. Using this terminology, we see the focus of developers on the production process.

Another group of people being primarily concerned with the meta process – especially in large organizations – might be a methodology group. 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 organization's projects in general.

If a company is on CMM level 5, continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies [20]. The organization as a whole and the projects themselves must address continuous realization of measurable software process improvement, like for instance defect prevention, technology change management, and process change management. Thus, the companies' methodology group must be able to improve and evolve the standard software process. Therefore this group needs a common Process Framework capturing the basic concepts of software development process models. Figure 1 illustrates three levels of an overall model for software development processes where the aforementioned views can be mapped on.

The Instance Level in Figure 1 captures those elements that belong to a certain project, such as an analysis document of a concrete project.

The Model Level describes a certain software development process. This process definition contains an outline of an analysis document or a description and guidelines of how to organize and hold a workshop with customers to elicit the requirements. This level offers the guideline and information 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 our Model Level.

The Metamodel Level provides the basic framework to establish a living process model. It offers clear definitions for terms like „Work Product" or „Activity". The Metamodel Level represents the common conceptual base of a company's methodology group to improve and evolve the underlying standard software development proc-

ess[2]. It is on this level where (the concepts of) process modeling languages, such as EPOS SPELL and SOCCA (cf. [9, 28]) are defined.

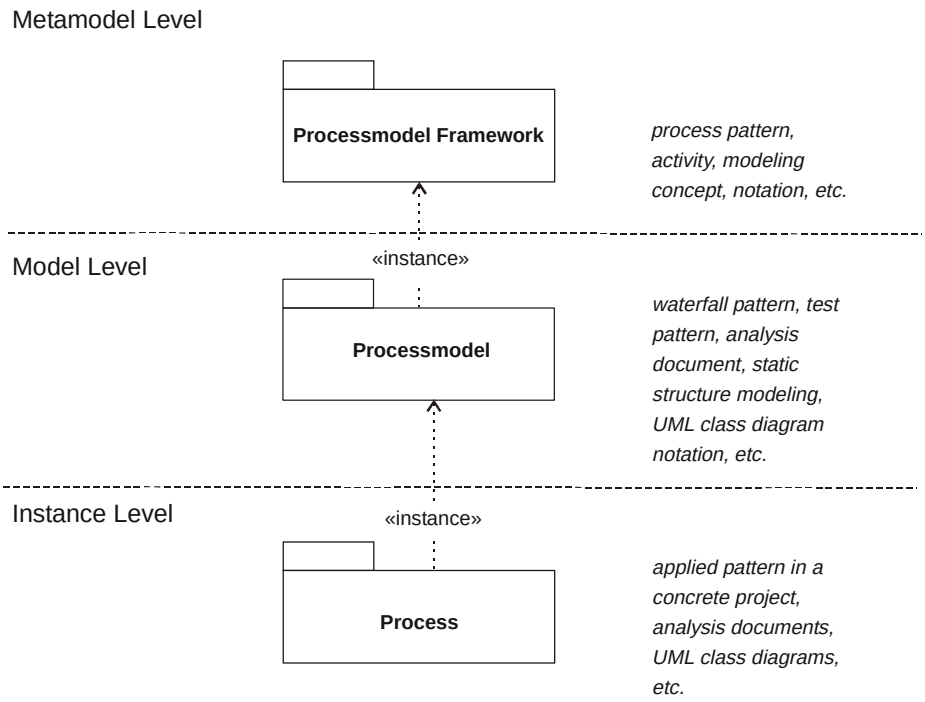

Fig. 1. The Layers of an Overall Process Model

2.2 The Project View of the Living Software Development Process

Project managers and project team members follow the concepts, guidelines, and help provided by a specific software development process defined on the Model Level in Figure 1. While performing their daily tasks they are creating instances on the Instance Level in Figure 1.

Managing a concrete project implies selection of a suitable process from a set of existing, possibly standardized alternatives. Then the chosen process has to be tailored accordingly to the project's characteristics. This tailored process represents the guidelines, which are to be followed in the project. In terms of our Process Framework, given in section 3, the tailored process defines which work products are to be pro-

[2] Note, this metamodel structure follows the guidelines provided by the Meta Object Facility (MOF) specification of the Object Management Group (OMG) [19].

duced, and which modeling concepts, notations, and patterns may be applied in the project.

2.3 The Method View of the Living Software Development Process

Process improvement, as required on CMM level 5 [20] for example, means the evolution of process models, i.e. of elements on the Model Level in Figure 1. The formulation of process models on the basis of a well-defined ontology facilitates comprehension and hence changes of development processes. Elements of the Metamodel Level in Figure 1 are supposed to play the role of such an ontology defining terms like “Activity”, “Process Pattern”, “Work Product” and their inter-relations.

An ontology for development processes provides both, 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 former process elements. On the basis of these proposals redefinitions by the methodology group can be done. Figure 2 shows this method view on a living software development process.

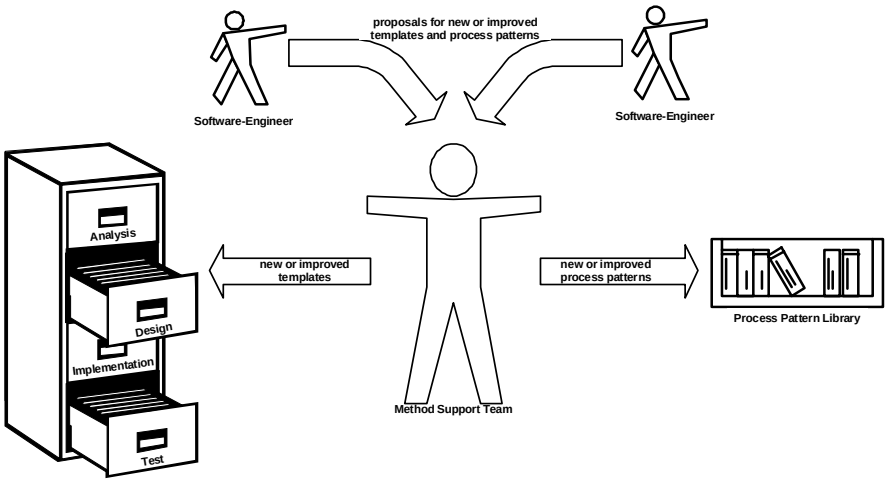

Fig. 2. The Living Software Development Process from the Method View

In our ontology, which we call the Process Framework, given in section 3, we follow the principle of separation of concerns so that changes are facilitated because of having minimal and localized effects.

3 Framework of a Living Software Development Process

In the previous section we have shown how developers and 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 Framework, which is on the Metamodel Level in Figure 1.

The Process 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. Additionally, the Process Framework must support static as well as dynamic adaptation of the process model with respect to the evolution and learning of a living organization (c.f. Section 1).

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 models 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 3 illustrates the basic concepts of the proposed Process 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.

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.

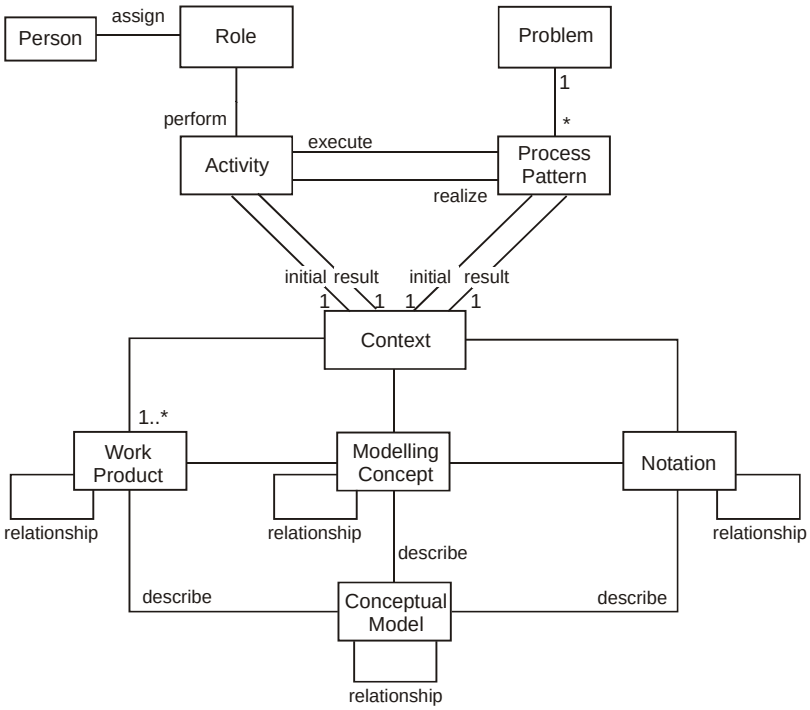

Fig. 3. The Process Framework

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 3 we require certain relationships between the contexts of related process patterns and activities. The work prod-

ucts in the result context of a Process Pattern have to be a superset 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.

An example of a realizes-relationship is shown in the context of a Business Process Modeling example in Figure 4. The activity Identify Business Tasks might be realized by the process patterns Inspecting Legacy Systems or Involving Business Experts respectively. The process pattern Inspecting Legacy Systems is a realization of the activities Identify Business Tasks as well as Refine Task Characteristics.

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 of the initial contexts of the process pattern they are executed by. Thus intermediate results produced in the workflow of the executed activities need not necessarily be part of the initial context of the executing activity. For an example of the executes-relationship in the context of Business Process Modeling we refer to Figure 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 3, such as work product and conceptual model, cover the general idea of structuring these elements, for example hierarchically.

4 Process Patterns

A process pattern enables us to describe and document process knowledge in a structured and well-defined way. As already discussed in Section 3 and illustrated in Figure 3 process patterns consist mainly of a problem solved by a proven solution applied through certain activities, and an initial as well as a result context. The activities play important roles in process patterns, because they reflect the tasks, which have to be carried out as described by the solution.

Process patterns, as well as all kinds of patterns, must be presented in an appropriate form to make them easy to understand and discussable. In this section, we first present a uniform pattern description template for process patterns. Then we provide a sample process pattern to illustrate the basic concepts of process patterns.

5.1 Pattern Description Template

A good description helps us grasp the essence of a pattern immediately – what is the problem the pattern addresses, and what is the proposed solution. A good description also provides us with all the details necessary to apply the pattern and to consider the consequences of its application. Moreover a uniform, standardized description template for process patterns is needed. This helps us to compare patterns, especially when we have to select from alternative solutions. For activities similar templates are useful. For brevity we omit a detailed description of this kind of template. Table 1 and Table 2 illustrate our proposal for an activity and process pattern description template by example. A detailed discussion of a wider range of process patterns is not in the scope of this paper and can be found in [7, 8]. Please note, that the sample Process Pattern of this section resides on the Instance Level of the Overall Process Model shown in Figure 1.

The example pattern template given in Table 2 contains, besides others, the following fields:

- Intent: A concise summary of the pattern's rationale and intent. It mentions the particular development issue or problem that is addressed by the pattern.
- Problem: The problem the pattern addresses, including a discussion of the specific development task, i.e. the realized activity, and its associated forces. Moreover the problem description may contain information with respect to consumers, competitors, and the market situation.
- Solution: A solution may suggest certain activities to be applied to solve a certain problem. Possibly an order may be given in which to perform these activities, or alternatives may be proposed. Besides that the solution comprises methodical guidelines and concrete recommendations. A solution shows a possible answer to balance the various forces that drove the project into the current situation. The solution includes a list of activities for execution. In contrast to these activities, the activity realized by the process pattern is referenced below. Moreover the solution depicts the relationships of the initial and result contexts of the executed activities and shows how the activities are combined.
- Realized Activities: The name of the activities for which the pattern provides a strategy of how to execute it. Every process pattern realizes at least one activity.
- Initial Context The internal state of the development project, i.e. the state of the corresponding work products, that allows the application of this process pattern.
- Result Context: The expected situation after the process pattern has been applied, i.e. the resulting state of the work products.
- Pros and Cons: A short discussion of the results, consequences, and tradeoffs associated with the pattern. It supports an evaluation of the pattern's usefulness in a concrete project situation. The problem description together with the pros and cons of a pattern helps us in choosing from alternatives, that is static and dynamic tailoring. Thereby these two pattern elements have a purpose similar to selection guidelines in [22].

- Example: Known uses of the pattern in practical development projects. These application examples illustrate the acceptance and usefulness of the pattern, but also mention counter-examples and failures.
- Related Patterns: A list of related patterns that are either alternatives or useful in conjunction with the described pattern.

In our example we chose the activity Business Process Modeling and a process pattern called Business Process Modeling Task Analysis with Activity Diagrams. This process pattern provides project team members with a strategy for developing a Business Process Model from an Initial Customer Specification.

Table 1 gives the description of the Business Process Modeling Activity:

Table 1. Activity Description: Business Process Modeling (BPM)

Entry	Activity Description
Name	Business Process Modeling (BPM)
Role	Business Expert, Software Architect
Development Issue	The goal of performing this activity is to develop a (complete) business process model, which covers all the business scenarios and business entities described by example in the Initial Customer Specification serving as input. Thereby the project goals, system vision, and constraints specified in the initial customer specification are to be taken into account.
Initial Context	Initial Customer Specification
Result Context	Initial Customer Specification, Business Process Model

Table 2 gives the description of the pattern BPM Task Analysis with Activity Diagrams, which realizes the Business Process Modeling activity:

Table 2. Process Pattern Description: BPM Task Analysis with Activity Diagrams

Entry	Process Pattern Description
Name	BPM Task Analysis with Activity Diagrams
Keywords	Business Process Modeling, UML Activity Diagrams, Stepwise Refinement, Iterated Modeling with Reviews
Intent	Development of – a precise and unambiguous documentation of a BPM – documentation of BPM on different levels of abstraction cover not only all details but also provide an overview of relevant business processes. – documentation of BPM such that it can be understood by business experts as well as software developers – ensured adequacy of BPM (validated model)
Problem	Business experts are available but a precise documentation of as-is and to-be business processes being relevant for the system to be developed does not exist.

	High complexity of business processes. The system vision of the initial customer specification hints at a strong relationship between the system to be developed and the business processes (e.g. support of large parts of business processes by the intended software system).
Solution	In order to achieve a precise and unambiguous documentation of business processes use UML activity diagrams [26] with its formal syntax to describe business processes. In order to achieve a documentation of different layers of abstraction apply the principle of stepwise refinement. Start with the definition of major tasks and refine them iteratively. Ensure adequacy of the business process model by reviewing each iteration of the model with (third party) experts. Involve business and software architecture experts being fluent with activity diagrams. The patterns workflow - namely its executed activities - is the following[3]: After having identified major tasks and user classes assign user classes as actors to tasks. Refine task characteristics of major tasks, and define a first version of the tasks' task chains by decomposing it into sub-tasks, and defining their causal dependencies. Consider alternative chains. Review this first model involving persons representing the identified user classes in the review. Perform the refinement steps of tasks iteratively and review each iteration (apply pattern Refinement of Activity Diagrams).
Realized Activities	Business Process Modeling
Initial Context	Initial Customer Specification with arbitrary modeling concepts and notations
Result Context	Business Process Model with UML activity diagrams as notation for task chains and a pre- and post-condition style specification of tasks.
Pros and Cons	Pros: – UML Activity Diagrams provide a standardized, concise and unambiguous notation to document business processes (Task Chains). The applied modeling concepts are widely used by business experts (e.g. similarity with Event Driven Process Chains) [24] as well as software developers (e.g. similarity with Petri nets). – This precise way of description supports detailed and precise review.

[3] The temporal ordering of the activities, which are executed by a Process Pattern, may be described by an UML Activity Diagram.

	– Iterative modeling and review increases understanding and quality of the business model. Cons: – Usage of specific notations, namely UML activity diagrams, may require training of involved persons. A common understanding of the notation must be ensured. – Stepwise refinement is a pure top down approach so that consideration of existing parts may be difficult.
Related Patterns	See also: BPM Informal Task Analysis

Besides BPM Task Analysis with Activity Diagrams, as mentioned in Table 2, the further ways for performing activity Business Process Modeling might exist. For example the process pattern BPM Informal Task Analysis represents an alternative strategy for business modeling proposing an informal documentation of business processes. This might be suitable when business processes are simple and the software system does not play a major role in these processes. The pattern map given in Figure 4 shows these two alternative performance strategies for the business modeling activity.

5.3 Managing the Process Lifecycle – Process Pattern Maps

As already mentioned process patterns will exist on several levels of detail. Like in other pattern-based approaches, a single pattern may be combined with others, forming the lifecycle of a development process. This lifecycle will not be fixed, but will vary from project to project and from situation to situation, according to the specific problems of the projects.

The selection of a process pattern may be outlined as follows: Based on the project's current situation, as partly represented by the state and consistency of work products, the project leader tries to identify the next activities he wants to be executed. This information leads to the selection of one or more alternative process patterns with initial contexts and problem descriptions matching the current situation. After a careful consideration of the alternatives' pros and cons and their problem descriptions one pattern is chosen. This pattern recommends a number of development activities and their temporal order. For each of its activities the solution may require or propose the application of certain process patterns.

By choosing process patterns the project manager forms the process lifecycle. Usually a process pattern library will contain a large number of patterns. We introduce two kinds of pattern maps providing an overview over a set of patterns by structuring them from different points of view.

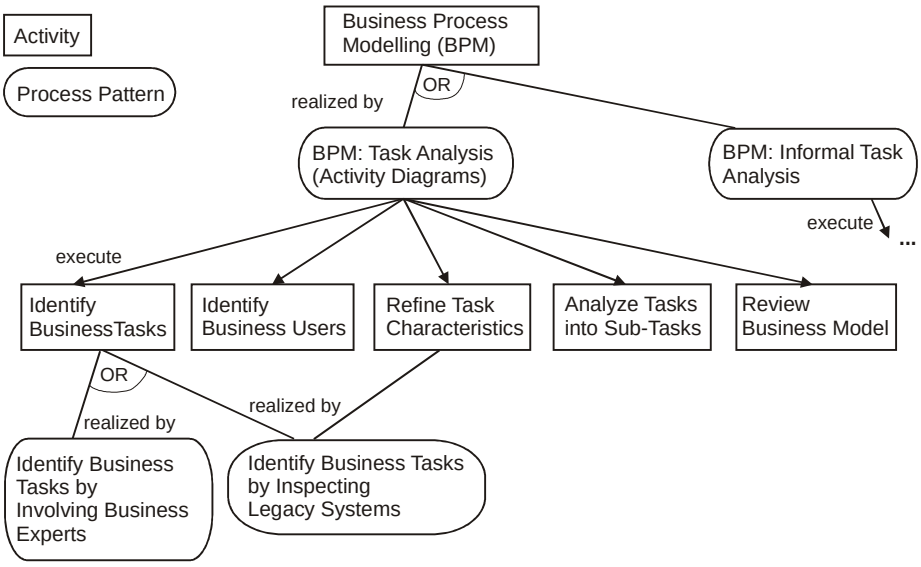

Fig. 4. An Activity Process Pattern Map

One possibility is to structure process patterns accordingly to the activities they realize. We call this activity process pattern map. Activity process pattern maps are directed graphs. These graphs have two kinds of nodes, namely activities and process patterns. A process pattern node has edges to all the activities that have to be performed by applying the pattern and exactly one edge to the activity it realizes. Each activity may be performed following the solution provided by a process pattern. Hence each activity node has edges to process pattern nodes that provide guidelines to perform the activity. Figure 4 illustrates such an activity process pattern map that builds a graph.

A second viewpoint on a set of pattern maps is the so-called context process pattern map. This kind of map is a directed graph with contexts as nodes and patterns as arcs. This way we can easily see alternative ways, i.e. process patterns, from one context to another. These maps are similar to the maps presented in [22].

6 Conclusion and Further Work

In this paper we enhanced existing Process Frameworks comprising elements, such as work product, activity, and role, by introducing the notion of process pattern as a modular way for the documentation of development knowledge. According to the best of breed idea, a company can combine the most appropriate solutions from different established methods and best practices to form a perfectly tailored process for a company and for all types of projects. Therefore we introduced a Process Framework that facilitates integration, tailoring, and evolution of process models.

Similar to the approach presented in [22], by stating the tackled problem as well as discussing pros and cons, patterns support selection of adequate strategies during process enactment that is problem-driven dynamic process tailoring.

A difference to other existing approaches is the explicit modeling of relationships between work products, modeling concepts, and notations, allowing us to describe and integrate generic processes, referring to work products only in general, with specific development processes providing concrete modeling concepts and notations.

To realize a process pattern approach within a company software developers need some guidance to find their way through the vast number of process patterns that may have been developed at their company. Moreover due to the continuous evolution and change of a living development process patterns a tool to store, present and manage process patterns and work product definitions would be very desirable. We are working on a tool supporting process model maintenance. By realizing the presented pattern maps this tool is supposed to provide guidance for software developers in finding their way through the jungle of process patterns.

To sum up the application of a process pattern approach seems to be very promising, as it provides a flexible way to define a tailored development process that can be easily adapted to new requirements. Combined with a reasonable tool support for the management and development of process patterns this approach may help organizations to create and evolve their custom development process.

References

1. Dirk Ansorge, Klaus Bergner, Bernd Deifel, N. Hawlitzky, C. Maier, Barbara Paech, Andras 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.
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.