

Support for the Process Engineer: The Spearmint Approach to Software Process Definition and Process Guidance

Ulrike Becker-Kornstaedt⁺, Dirk Hamann⁺, Ralf Kempkens⁺,
Peter Rösch⁺, Martin Verlage⁺, Richard Webby^{*}, Jörg Zettel⁺

⁺Fraunhofer Institute for Experimental Software Engineering (IESE),
Sauerwiesen 6, D-67661 Kaiserslautern, Germany
{becker, hamann, kempkens, roesch, verlage, zettel}@iese.fhg.de

^{*}Center for Advanced Empirical Software Research (CAESAR), School of Information Systems
The University of New South Wales, Sydney 2052, Australia
r.webby@unsw.edu.au

Abstract The software development process and its related activities are described, implemented, analyzed, and changed by so-called Process Engineers. Process Engineers provide descriptions of software development processes to Process Performers. Because the processes usually are complex, support is needed for both Process Engineers and Process Performers. This paper reports the development and application of the process modeling environment Spearmint¹. The architecture of Spearmint allows for a flexible definition and addition of views which are used for retrieving filtered and tailored presentations of the process models. One distinct view, namely the Electronic Process Guide used for dissemination of process information and guidance of Process Performers, is discussed in more detail. The Spearmint environment has been validated in industrial process engineering cases.

Keywords: Process Engineering, Process Guidance

1 Introduction

Besides product development, software engineering includes engineering of the software development process. Process improvement and process programming, deal with explicit process representations (i.e., *process models*) in order to analyze process behavior, guide Process Performers, enforce rules, or automate process steps. The process model is the basis for specifying how the process is carried out. It is a vehicle for enabling better understanding and communication of software processes. Various research groups and workshops are addressing process modeling issues under terms

1. Spearmint is a registered trademark of Fraunhofer IESE, Kaiserslautern, Germany.

like process-centered software engineering environments and software process management environments.

A *Process Engineer* is responsible for eliciting process knowledge from experts, capturing the processes and process steps in a process model, analyzing both the process models and the real process, disseminating process changes, and for implementing systems to provide automated support to Process Performers. From our experience gained in industrial process modeling case studies, we have found that a typical software process will require modeling up to a hundred entities: process steps, documents, and roles [6]. This cannot be achieved without a well-defined set of guidelines, concepts, and tool support for the Process Engineer.

Software development processes are performed by a large number of people. These processes are very unlikely to be as 'straightforward' as existing process models, for example, the waterfall lifecycle model [6]. Analysis of real software development processes has uncovered, for example, that control flow in software development projects is complicated [16]; the number of process steps high and they are strongly interrelated by means of product flow and control flow [17]. Also, personal interpretation of official process documents may lead to process variants and inconsistent performances by different people [4]. Therefore, support for Process Performers is crucial for the coordination of software development processes.

Most work on support for Process Performers has focused on automated support of the software process ('*enactment*'). Many approaches for the support of Process Performers are influenced by workflow systems. However, experience from industrial projects indicates that automation is only achievable for fine grained, tool-related activities [1,10]. In the context of real-world industrial projects, the understanding of processes, communication and their analysis have much more relevance.

We investigated the capabilities of a number of existing approaches, but found that they all had significant limitations for the purpose of providing practical support for the Process Engineer. This was one major reason why we decided to develop a new approach which we called **S**pearmint (**S**oftware **P**rocess **E**licitation, **A**nalysis, **R**eview, and **M**easurement in an **I**N**T**egrated Modeling Environment) [19]. Spearmint provides an integrated modeling environment using a visual domain-specific process modeling language which is used to describe and define real-world software processes. Among others it supports the generation of a browseable process description (Electronic Process Guide) which is a support tool for Process Performers when performing their tasks.

This paper is structured as follows: Section 2 clarifies tasks and requirements for process engineering. In Section 3 we introduce Spearmint. Section 4 gives an overview of related work. Section 5 discusses the results gained in the application of Spearmint. Section 6 concludes with a summary and an outlook to future work.

2 Support for Process Engineering

The role of a Process Engineer is ideally performed by someone outside of the development team. This may be a member of a software engineering process group

(SEPG) [11], or the Project Manager may be given a set of additional tasks involved by this role, or it may simply refer to a portion of the quality management in an organization. In any case, the Process Engineer deals with more or less formal descriptions of the process, transforms and manipulates them, analyzes them, and packages them for use by other people in his organization. The most relevant tasks the Process Engineer is responsible for are the following (compare also to [15] and [11, page 290]):

- elicitation of process information from both humans and existing process documentation,
- definition of the process as a process model using more or less formal approaches,
- analysis of the process model to check for consistency and dynamic properties,
- design of process models carrying the results of a process change,
- implementation of the process either as a process program or by definition of organizational mechanisms (e.g., measurement forms),
- provision of process models to Process Performers for guidance or enforcement.

To perform these tasks, different approaches to define a process are needed. For example, descriptive process models are used to capture the actual software development process in an organization, whereas prescriptive process models, like standards and guidebooks, require a different approach to the modeling of processes. Using a fine grain process programming language to implement process fragments that are executed or interpreted by a process engine requires other capabilities from a Process Engineer than when re-designing an existing process to reflect changes. Process engineering is often aggravated by the complexity of real-world software processes – and consequently the complexity of their process models.

3 The Spearmint Environment

Spearmint is an integrated environment for modeling, analyzing, and measuring processes (<http://www.iese.fhg.de/spearmint>). It supports Process Engineers during elicitation, editing, review, and analysis of process models by providing navigation and abstraction support. Internet-browseable views, which constitute the Electronic Process Guide (EPG), for guidance of Process Performers can be generated easily from Spearmint process models. The conceptual schema of Spearmint was influenced by existing approaches (e.g., [2]) as well as experience gained in industrial process modeling case studies.

Experiences from industrial process modeling cases formed the driving forces behind identifying the major requirements for our tools: a comprehensive conceptual schema, a graphical and easily understandable user interface, consistent management of different views of process models, a graphical notation, and the possibility to provide a support for Process Performers during execution.

Section 3.1 introduces a number of requirements for tools aiming at support for process engineers. Section 3.2 describes the main conceptual and user interface aspects of the Spearmint modeling environment. Section 3.3 explains the EPG in more depth. Section 3.4 discusses the integration of the modeling environment and the EPG.

3.1 Views on Process Models

A means to manage the inherent complexity of models of real-world software processes is to divide a process model into *views* [17]. In general, views on a process model can be seen as overlapping subsets of the overall information represented in a process model. Views might be used to concentrate on certain aspects of a process, like control flow, or work breakdown structure (i.e., process hierarchy). Role-specific views, for example, may show only a subset of the process model, namely exactly the information relevant to a particular role. For the Process Engineer, the usage of role-specific views allows to have less complex process models to deal with. For the Process Performer, role-specific views aid in understanding, since they focus only on those parts relevant to the role. Conceptually, we consider a view on a process to be comprised of five characteristics:

- *Objects*: the data to be used for presentation, e.g., activities, artifacts, or roles. A schema is used to represent the objects and their relationships [18].
- *Aspect*: the portion or slice of the process model selected for representation, e.g., product flow, or decomposition. The aspect can be expressed as a subset of the process model schema which is used to represent the objects.
- *Style*: the way in which the data is represented to the user. For the same data set, different representation styles are possible, e.g., product flow between activities and artifacts could for instance be represented using a diagrammatic representation or a table.
- *Synthesizer*: set of functions over the object types that transform the objects for purposes of the required abstraction level. An example for a synthesizer is summarizing effort data from sub-activities into the data of a compound high-level activity.
- *Mechanism*: set of procedures to modify the representation that can be invoked by the user, e.g., simple cut/copy/paste services.

Aspect, style, synthesizer, and mechanism describe a *View Type*. A view type is a general description of what type of objects should be presented to the user, and how they should be presented. View types are instantiated by assigning objects to them.

Process Engineers may use process modeling environments to create and maintain process models. In addition to general requirements (e.g., domain specific schema) which are listed in [9] we consider the following requirements as important in order to cope with process models in industrial environments. These requirements explicitly take into account the use of views.

R1: The process modeling environment must provide and manage different views of a process model. Views filter information (e.g., by not showing all objects) or make it more dense (e.g., show more abstract artifact types than actually accessed by the activity displayed). Hence they reduce complexity.

R2: The process modeling environment must offer predefined view types as well as services to define view types. There is no best set of views available. In our experience it is likely that a new view type is required (e.g., display the relationship between roles and artifacts) when dealing with a new problem.

R3: The process modeling environment must provide concurrent updating mechanisms of objects to keep consistency across views.

R4: The process modeling environment must provide visual cues to track relationships among views, for example, marking process elements in all other views when selected in one view only.

R5: The process modeling environment must provide mechanisms to generate process descriptions, i.e., to export views which are used for guidance of Process Performers.

3.2 Conceptual Schema and Modeling Environment

Spearmint is a new development concentrates on mechanisms to provide different views of a process model to Process Engineers and Process Performers. Spearmint is based on a domain-specific and canonical conceptual schema of process model elements [18]. The most important elements of this schema, the user interface to access process models, and the architecture of Spearmint will be described in this section.

The most important information units needed to describe a real-world software process are mapped onto the following elements of the conceptual schema:

- Artifacts are abstractions of any documents or products that are accessed in a project, as a desired or intermittent result of the project or as an input to an activity in some other form.
- Activities are process steps which may cover software development and maintenance activities as well as project management or quality assurance. Activities consume and/or produce artifacts (product flow).
- Roles are abstractions for a set of responsibilities or skills necessary to perform an activity.
- Tools represent computer programs, or other means that are used to support or automate an activity.

In addition to the concepts mentioned here, the comprehensive conceptual schema contains entities, such as organization, or measurement concepts. (They are discussed in detail in [18]).

The schema is used to define the structure of the *comprehensive process model* which is the union of the objects of all views [17]. The views are used for creating, modifying, and displaying the comprehensive process model or subsets of it. For example, the Process Engineer creates a new, empty part of the comprehensive process model by instantiating an artifact decomposition view type in order to first enter the hierarchy of all artifacts. Whenever he modifies the view displayed (i.e., adding, or renaming an artifact, or specifying an aggregation relationship between artifacts), the comprehensive process model is updated.

The number of concurrently existing views is limited only by available system resources. All views are kept consistent automatically by the system. The view types defined so far in Spearmint are:

- *product flow view*,
- *properties view*,
- *decomposition view*, and
- *textual view*.

The product flow view is a graph-like interface to the process model, which uses simple icons for entities, connected by lines for relationships. Figure 1 shows an example product flow view. It contains an activity *Implement*, the artifacts used in that activity, a role and a tool assigned to that activity.¹

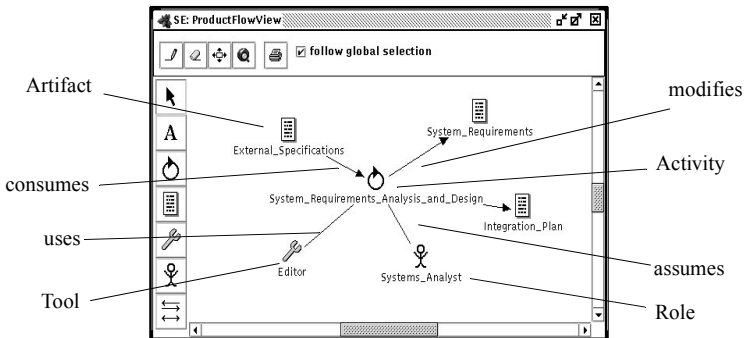


Figure 1. Spearmint: Product flow view

Details of a process model element can be entered using a properties view. This is a dialog, in which general properties like name and description of the element can be edited (see Figure 2). It also supports the flexible definition of *attributes*. Attributes can be dynamically added to and removed from an element. If used in the context of measurement, *values* can be associated with the attributes. In order to better be able to consider the needs for EPGs the Spearmint modeling environment has an extra set of EPG attributes which can be attached to activities, artifacts, or roles. These attributes allow for instance to describe the detailed steps of activities, or to attach links to template documents to artifacts.

Modularity, as a concept for structuring a process model, is supported by the decomposition view (Figure 3). This view allows the user to define and browse the hierarchical decomposition of artifacts, activities, and roles in a representation style familiar to him. The textual view is strongly related to the Electronic Process Guide and will be discussed in Section 3.4.

1. This view can be described as follows: Objects = {System_Requirements_Analysis_and_Design, External_Specifications, System_Requirements, Integration_Plan, Systems_Analyst, Editor}; Aspect = {Activity, Artifact, Role, Tool, Product Flow, Role Assignment, Tool Usage}; Style = {Artifact = [document icon], Role = [stick figure icon], Activity = [flame icon], Tool = [wrench icon], Product Flow = [dashed line icon]}; Synthesizer = <none>; Mechanisms = {Create Activity, Create Artifact, Create Tool, Create Role, Link, Delete, Cut, Copy, Paste, Print}

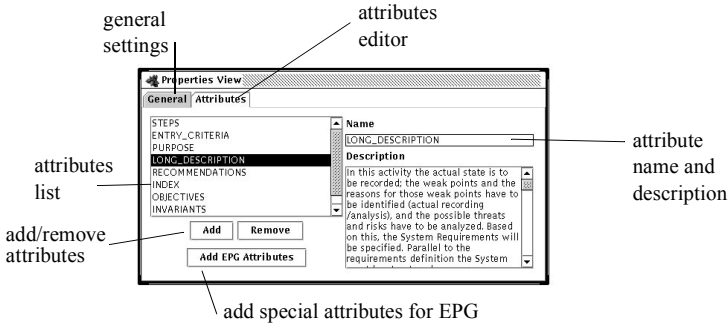


Figure 2. Spearmint: Properties View

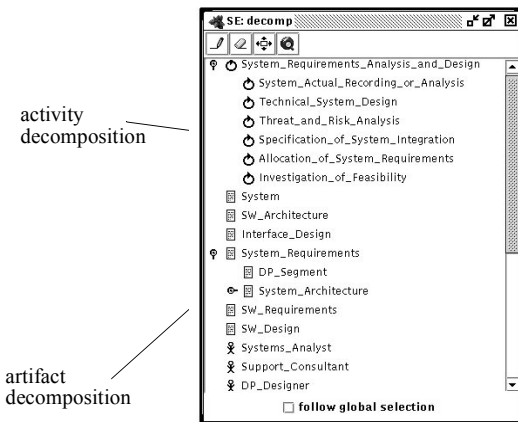


Figure 3. Spearmint: Decomposition View

The implementation of Spearmint uses Java as the technological basis. The integrated software architecture of Spearmint consists mainly of three layers: the Process Model Layer, the Partial Model Layer, and the User Interface Layer. Figure 4 explains the interplay among these in detail.

The bottom layer (Process Model Layer) contains the comprehensive process model, i.e., all process model elements, their attributes, and relationships. This layer is independent from the notation chosen in a specific diagram and forms the canonical basis for all views. The Process Model Layer stores process models using the schema introduced above. In the example used here, the Process Model Layer would comprise the activity *Implement*, the role *Author*, the tool *Editor*, the artifacts used in the product flow and all the relationships among these, and the attribute *Effort* (depicted in Figure 2). We are currently using ObjectDesign/PSE, a simple public-domain file-based object-oriented database, to implement data access services to that layer. This can easily be upgraded to the full functionality of the ObjectStore database system.

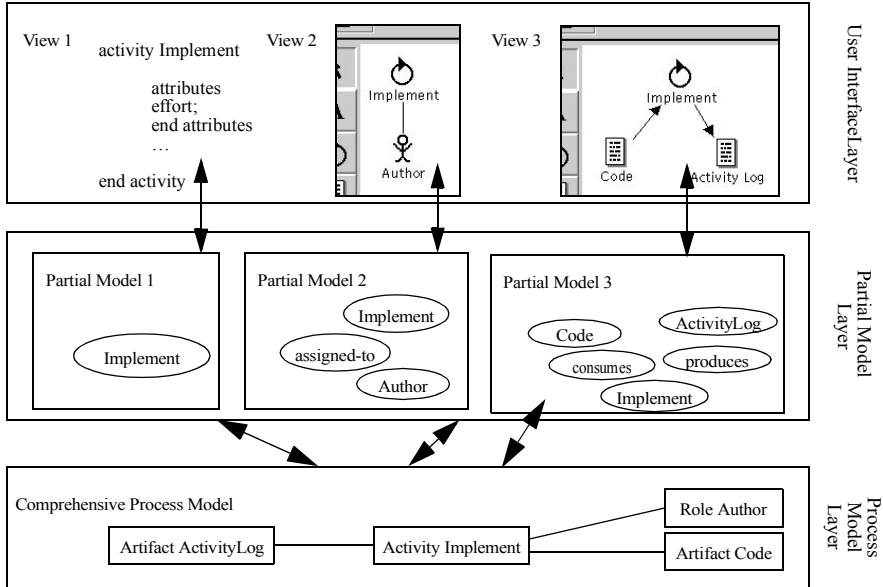


Figure 4. View Management in Spearmint

The Partial Model Layer bridges the gap between the user interface and the comprehensive process model (i.e., Process Model Layer). It is responsible for managing the objects and aspects of a view. They are subsets of the comprehensive process model. The Partial Model Layer translates process model elements into objects, which are more convenient for supporting a specific notation. Therefore this layer has to perform transformations like composing multiple objects into single container objects. In the figure, for example, Partial Model 2 contains an activity *Implement*, the role *Author*, and the *assigned-to* relationship between them. The schema to store the objects is the same as for the comprehensive process model.

The User Interface Layer performs the visualization of process models. The User Interface Layer creates views by assigning view types to Partial Models, manages the views, and provides the user interface. According to the information in the view type the objects of the Partial Model Layer are mapped onto symbols of the presentation which are displayed to the Process Engineer. In addition to the objects and aspects provided by the Partial Model Layer, the User Interface Layer defines the style in which the information is to be presented and provides transformation functions over the data types (synthesizer) as well as modification procedures (mechanism). Each view in the User Interface Layer is related to one Partial Model. In our example, View 2 displays the objects contained in Partial Model 2, but in a user-defined iconic notation, and provides editing services to modify the representation.

Spearmint maintains the consistency between views, Partial Models and the comprehensive process model automatically. This is the basis for a rich set of powerful interactive features at the user-interface level, which are described in more detail in [19]. This architecture allows easy addition of new view types. We found this kind of flexi-

bility very important because in process engineering tasks tailored representations were required (e.g., when Process Performers were accustomed to a particular representation style).

What view types are needed for what process engineering situations is subject to further investigation. Experience with the tool in industrial improvement programs will therefore be incorporated into future increments of the prototype environment.

3.3 Electronic Process Guide



An important process engineering activity is the dissemination of process knowledge. An Electronic Process Guide (EPG) aims to support Process Performers [13]. Typical problems occurring during process performance are related to one of the following:

- staff turnover is high, new team members must become familiar with the process,
- team members must perform activities which they may be unfamiliar with, which may be infrequently performed, or involve many interrupts and context switches,
- communication is impeded due to large development teams or different geographical locations.

In order to overcome these problems and to provide concrete guidance, process-relevant information has to be made accessible to Process Performers in a way that is easy to use for them. In contrast to approaches which use an explicit enactment mechanism and a process engine, the EPG gives *guidance* to the user, that is, it provides the Process Performer with information about the actual state, history, context, and future steps of the process to make informed decisions. The user himself decides what information to access and at what level of detail. This complies with [1] and [10] which advocate that strict enforcement of a prescribed process imposed by many of the enactment mechanisms is not adequate for all tasks in software development.

The technical implementation of the EPG is based on Web-technology. On the one hand this provides a tool and appearance which is already familiar to Process Performers and does not involve a huge investment in new tools. On the other hand this allows to benefit from already existing browser features, like setting bookmarks, or using hyperlinks. In addition to being a web-based process guidebook the EPG provides services like managing personal annotations or links to example and template files. A computer-based guidebook allows fast updating and therefore ensures that Process Performers use a consistent version of the process handbook and the process information needed. Typical usage scenarios for an EPG are:

- A Process Performer familiar with the process needs help in unexpected situations or complex or infrequently performed activities. The EPG may provide additional information, such as manual pages, links to example or template files.
- A novice who joined the team recently needs to become familiar with the most frequent activities and artifacts. The EPG would enable him to navigate and explore the process by following the links and relationships.
- Process performers working at different sites are expected to use a consistent definition of the process. Updated process descriptions can be easily provided even to different geographical sites.

Activities, artifacts, and roles are described on *main pages*. Figure 5 depicts such a main page for the activity *Implement* generated from the example shown in the previous section. The graphical decomposition frame (top left) shows the position of the activity within the decomposition tree. This interactive graphic facilitates navigation. The description (on the right) provides the main information, such as artifacts used. In the description glossary information is integrated as tool-tip information, i.e., when the user rests the mouse pointer on the -icons. Clicking on the on the -icon navigates the user to a whole page providing glossary information. The overview section (bottom left) provides access to the description section via links. In order to provide supplementary information within the same window, the glossary (on the bottom) contains brief descriptions of items referred to in the description section, for instance the artifacts used. Main pages for artifacts and roles have the same structure.

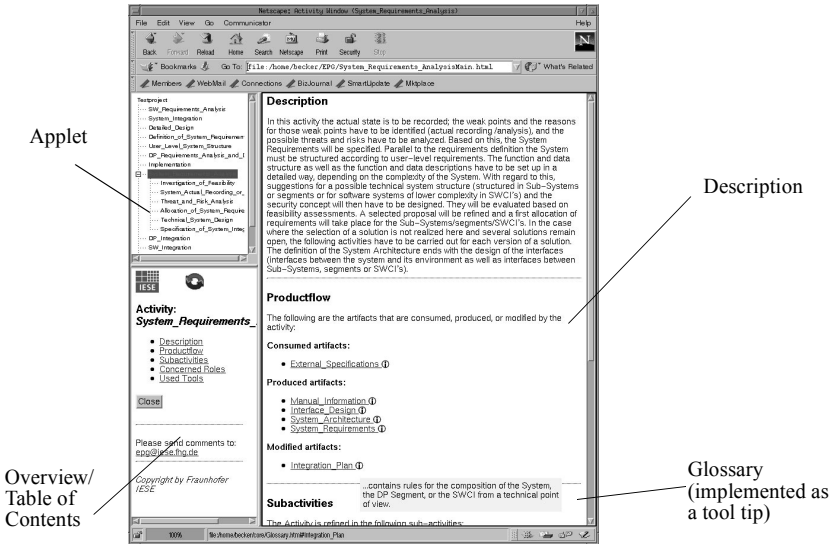


Figure 5. Screenshot of an EPG Activity page

3.4 Technical Integration of Spearmint and Electronic Process Guide

Web-technology used for the EPG allows for updates of the process model in a consistent and cost-effective way, even for multi-site projects and organizations.

We have developed an integrated architecture which addresses the needs of the modeling environment of Spearmint as well as those for the EPG. Figure 6 presents this architecture, showing how the potentially distributed tasks of process definition and process guidance access a common database. On the right side of the figure, a web server provides the EPG as HTML files which are generated from the Spearmint database. Whether the generation is done on a regular basis or dynamically on demand is

arbitrary and depends on the application context of the EPG. On the left side, Spearmint provides a special textual view which is a light-weight combination of an EPG generator and a web server. This view provides both an immediate update after process changes as well as navigation features between web browsers showing the EPG and views in the Spearmint environment. Thus the textual view is an excellent aid for the Process Engineer when discussing a process model with one or more Process Performers, or when guiding them through a process. In combination with a telephone, this can even be done over long distances and/or multiple sites.

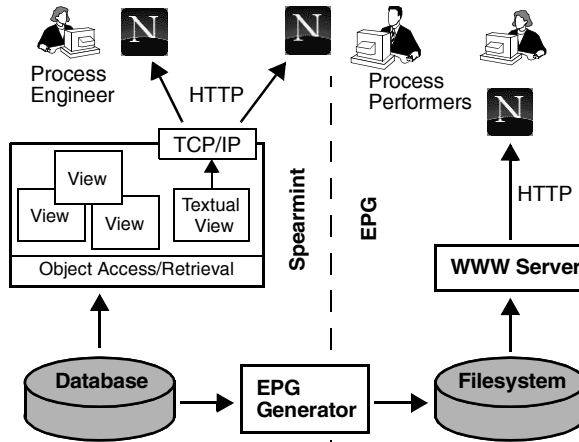


Figure 6. Spearmint and EPG System Architecture

4 Related Approaches

The Process Engineer may be supported by a number of different tools ranging from the very simple to the sophisticated:

- Graphic editors provide simple definition and placement of nodes and edges to create diagrams depicting properties of process models.
- CASE tools and meta-CASE tools (frameworks or generators to create CASE tools) may be used because they provide better drawing capabilities. Sometimes CASE tools have been extended to address process needs, mainly for business process re-engineering, can be also used for software process modeling.
- Simulators, like those based on the idea of System Dynamics [14], may be used to analyze dynamic aspects of processes.
- Specific product development tools, like STATEMATE [12], may be used to support one or a small set of tasks of the Process Engineer, like process analysis by simulating processes.
- Process-sensitive software engineering environments (PSEEs), like Process WEAVER [3], or Workflow Management systems provide some services to create

and manipulate process models. However, often the user interface does not support the Process Engineer adequately and comprehensively, and the focus is not on guidance but on automation.

- Process tools are designed for special tasks of the Process Engineer. For example, the tool Balboa [8] is designed to identify control flow patterns out of event data taken from the actual process performance.

Achievements have also been made in developing a common understanding about what elements should be covered by any of these languages (e.g., the schema by Armitage and Kellner [2]). What has not been tackled extensively in the past is the management of complex process models and especially support by multiple views.

5 Validation of Concepts and Prototype

Spearmint was used to capture a multi-site development process for the evolution of a large telecommunications product where more than 200 developers work on a base product and special extensions like applications for call centers. The process model was used within an overall measurement project to improve the process. The views were used to prepare role-specific views for review with interviewees as well as site-specific process models to cope with variants and interfaces between sites. The views reduced the model's complexity which in turn reduced effort for creating the model compared to former projects. An EPG has been installed recently at the organization's intranet. First feedback is very encouraging.

A major industrial validation of the Spearmint approach to descriptive process modeling was performed recently within the context of a software process improvement effort with an organization developing space software. The company wanted to develop a systematic measurement plan in order to improve its development processes. Based on role-specific interviews a process model was created. It contained some fifty activities, artifacts, and people responsible for the activities, plus the relationships among them. The view concept proved to be very helpful in reducing the complexity of the process model. The resultant descriptive model allowed us to recommend improvements to the current process, which are currently being implemented.

Spearmint and its predecessor MVP-E [5] have been used to formalize prescriptive processes as described in a variety of standards and guidebook, like the IEEE Standard 1074, ISO 12207. Our process modeling environment had been used to review a draft version of a national German standard. Based on the outcome of the consistency checks, major improvements were suggested which have been incorporated into the final version of the standard.

Spearmint was used to analyze the completeness of a real-time systems development process as described in a book in order to determine whether it is possible to create process models out of a technology description [5]. The case was based on [7] which documents the Structured Design Technique (SDL). The purpose was to transform the contents of the book into a formal process to check for consistency and completeness. Using the analysis functions, it was found that major information is lacking (e.g., documents are produced but never used).

The full text of the 1997 version of a national German standard, called ‘Das Vorgehensmodell’ (V-Modell), was prepared as an EPG and is available on-line (<http://www.iese.fhg.de/VModell>). In addition to the public web site, several copies of this instance of the V-Modell EPG are installed at German companies. Positive feedback from the users encourages us to further enhance and improve this technology.

6 Summary and Outlook

The Process Engineer needs support in managing complex process models. In this paper we presented the Spearmint approach which especially focuses on helping in creation of process models and their dissemination. Existing tools seldom address these two responsibilities of the Process Engineer.

The services provided by Spearmint stem from requirements which came up during industrial process modeling cases where tool support was missing. Commercially available tools have been found inadequate at including domain knowledge about software development processes. Especially they often lack sophisticated services to analyze process models and to check for consistency. Research prototypes in contrast are too specific in the tasks of the Process Engineer they support. Spearmint tries to bridge the gap between commercial process tools and research prototypes. In summary, the unique features of Spearmint are:

- A comprehensive domain-specific schema based on a combination of sophisticated approaches and experience gained in industrial process engineering tasks (e.g. [2, 9]). It also adds its own innovations - for example our schema allows the ability to define parallel abstraction hierarchies for multiple views [17].
- An architecture allowing rapid definition of new view types. Spearmint is not limited to predefined view types, but is easily extendible when the need for a new type of view becomes apparent.
- Concurrent updating of model information. A mechanism for change propagation sends model repository changes to all interested views.
- Visual cues aiding orientation in large process models.
- The possibility to generate EPGs which can be used by Process Performers to navigate through complex process information.

This paper has described how Spearmint supports the Process Engineer when creating a process model and disseminating process knowledge using Intranet technology. The future development topics to be addressed in Spearmint include further exploitation of Web-technologies and the improvement of the conceptual schema aimed at supporting process performance in a more explicit way.

Web-technology helps support the communication of process knowledge among the (potentially geographically dispersed) people performing and studying software processes. Web-based documents are easy to distribute among Process Performers and are accepted because they integrate well at the computer's desktop level. The process model is always up-to-date. Therefore it is more likely to be used than a heap of dust-covered documents on the shelf.

So far, interviews with Process Performers in industrial settings strongly support our preference to concentrate first on a visual, communicative and understandable representation of process models. Our second aim, which is to support process performance, is achieved by providing detailed descriptions of process elements which can be accessed using the EPG. In addition, we will implement services around the EPG like mechanisms for attaching annotations to elements of a process model or search [13] which will be used as a target for the HTML pages generated by Spearmint.

Further research is needed to fully understand the role of the Process Engineer and the potential for process technology such as Spearmint to support this role. Our future research will concentrate on additional view types and other user interface features for dealing with process complexity.

Acknowledgments

We would like to thank Marc Kellner, and Bill Riddle for very fruitful discussions about the EPG. We also thank Andrew Beitz, Lionel Briand, and Louise Scott for their comments, which led to significant improvements to the structure of the paper. We very much appreciate the work by all the students within the development of Spearmint.

References

- [1] Jim Arlow, Sergio Bandinelli, Wolfgang Emmerich, and Luigi Lavazza. A fine-grained Process Modelling Experiment at British Airways. *Software Process—Improvement and Practice*, 3(3):105–131, November 1997.
- [2] James W. Armitage and Marc I. Kellner. A conceptual schema for process definitions and models. In Dewayne E. Perry, editor, *Proceedings of the Third International Conference on the Software Process*, pages 153–165. IEEE Computer Society Press, October 1994.
- [3] Denis Avrillionis, Pierre-Yves Cunin, and Christer Fernström. OPSIS: A view mechanism for software processes which supports their evolution and reuse. In *Proceedings of the Eighteenth International Conference on Software Engineering*, pages 38–47. IEEE Computer Society Press, March 1996.
- [4] Sergio Bandinelli, Alfonso Fuggetta, Luigi Lavazza, Maurizio Loi, and Gian Pietro Picco. Modeling and improving an industrial software process. *IEEE Transactions on Software Engineering*, 21(5):440–454, May 1995.
- [5] Ulrike Becker, Dirk Hamann, Jürgen Münch, and Martin Verlage. MVP-E: A Process Modeling Environment. *IEEE TCSE Software Process Newsletter*, (10):10–15, Fall 1997.
- [6] Ulrike Becker, Dirk Hamann, and Martin Verlage. Descriptive Modeling of Software Processes. In *Proceedings of the Third Conference on Software Process Improvement (SPI '97)*, Barcelona, Spain, December 1997.
- [7] R. Bræk and O. Haugen. *Engineering Real-time Systems: An object-oriented Methodology using SDL*. Prentice Hall, New York, London, 1993.
- [8] J.E. Cook and A.L. Wolf. Balboa: A framework for event-based process data analysis. In *Proceedings of the Fifth International Conference on the Software Process*, pages 99–110, Chicago, IL, USA, June 1998. ISPA Press.

- [9] European Computer Manufacturers Association. Reference model for frameworks of software engineering environments. Technical Report TR-55, ECMA, 114 Rue du Rhone, 1204 Geneva, Switzerland, June 1993.
- [10] Volker Gruhn and Juri Urbainczk. Software process modeling and enactment: An experience report related to problem tracking in an industrial project. In *Proceedings of the Twentieth International Conference on Software Engineering*, pages 13–21, Kyoto, Japan, April 1998. IEEE Computer Society Press.
- [11] Watts S. Humphrey. *Managing the Software Process*. Addison Wesley, Reading, Massachusetts, 1989.
- [12] Marc I. Kellner. Software process modeling support for management planning and control. In Mark Dowson, editor, *Proceedings of the First International Conference on the Software Process*, pages 8–28. IEEE Computer Society Press, August 1991.
- [13] Marc I. Kellner, Ulrike Becker-Kornstaedt, William E. Riddle, Jennifer Tomal, and Martin Verlage. Process guides: Effective guidance for process participants. In *Proceedings of the Fifth International Conference on the Software Process*, pages 11–25, Chicago, IL, USA, June 1998. ISPA Press.
- [14] Chi Y. Lin, Tarek Abdel-Hamid, and Joseph S. Sherif. Software-engineering process simulation model. *Journal of Systems and Software*, 38(3):263–277, September 1997.
- [15] Jaques Lonchamp. A structured conceptual and terminological framework for software process engineering. In *Proceedings of the Second International Conference on the Software Process*, pages 41–53. IEEE Computer Society Press, February 1993.
- [16] Dewayne E. Perry, Nancy A. Staudenmayer, and Votta, Jr., Lawrence G. Understanding and improving time usage in software development. In Alfonso Fuggetta and Alexander Wolf, editors, *Software Process*, Trends in Software, chapter 5, pages 111–135. John Wiley & Sons, 1996.
- [17] Martin Verlage. An approach for capturing large software development processes by integration of views modeled independently. In *Proceedings of the Tenth Conference on Software Engineering and Knowledge Engineering*, pages 227–235, San Francisco Bay, CA, USA, June 1998. Knowledge Systems Institute, Skokie, Illinois, USA.
- [18] Richard Webby and Ulrike Becker. Towards a Logical Schema Integrating Software Process Modeling and Software Measurement. In Rachel Harrison, editor, *Proceedings of the Nineteenth International Conference on Software Engineering Workshop: Process Modelling and Empirical Studies of Software Evaluation*, pages 84–88, Boston, USA, May 1997.
- [19] Richard Webby, Peter Rösch, and Martin Verlage. Spearmint - a prototype tool for visualising complex software processes. In *Proceedings of the Third Biennial World Conference on Integrated Design & Process Technology (IDPT'98)*, volume 4, pages 297–304, Berlin, Germany, July 1998.