# MVP-E: A Process Modeling Environment

Ulrike Becker[1]
Dirk Hamann[1]
Jürgen Münch[2]
Martin Verlage[1]

[1]Fraunhofer Institute for Experimental
Software Engineering (IESE), Germany
[2]University of Kaiserslautern, Germany

**Ulrike Becker**, **Dirk Hamann**, and **Martin Verlage** can be reached at: Fraunhofer Institute for Experimental Software Engineering, Sauerwiesen 6, D-67661 Kaiserslautern, Germany; Email: {becker,hamann,verlage}@iese.fhg.de; http://www.iese.fhg.de/Competences/QPE/PM/.
**Jürgen Münch** can be reached at: AG Software Engineering, Postfach 3049, University of Kaiserslautern, D-67653 Kaiserslautern, Germany; Email: muench@informatik.uni-kl.de.

## 1. Overview

Software process models are a key for systematic software development. Project plans, guidelines, process standards, records, or process definitions are examples of, mostly informal, process models. Process models represent knowledge about organizational and technical processes explicitly. Furthermore, they make this knowledge persistent. This is needed in improvement programs where processes need to be documented, analyzed, changed, and implemented. At a more abstract level, process models can be used in two different ways: descriptively and prescriptively. Descriptively means that process models represent parts or aspects of an existing process in order to allow for better understanding. Prescriptive usage means that the process models define how a particular process should be performed.

We learned from a number of industrial projects that a formal language and specialized tools facilitate the development of consistent process models. Most of the approaches in software process technology did not fit our needs because they were designed to either study principles related to tools (e.g., integration) or the behavior of processes. A lot of existing systems focus on fine-grain, deterministic process models and their automatic interpretation which is contrary to our need of models on design level which allow for stochastic process performance.

This paper introduces the MVP-E (multi-view process environment) system. It illustrates how the tools developed for the process modeling language MVP-L support both descriptive and prescriptive usages of process models. All tools are based on the process modeling language MVP-L. This common representation schema facilitates the transition between creation and enactment of process models.

Section 2 gives an overview of MVP-E, the environment for process modeling and the language MVP-L for describing the processes is given. The discussion in Section 3 concentrates on the component MoST which is the central part of MVP-E. The usage of MVP-E is illustrated in Section 4 by discussing both industrial application and experimental studies of both prescriptive and descriptive modeling. Recommended additional support for process engineers, that is not yet covered by MVP-E, is discussed in Section 5. Finally, Section 6 gives a brief summary and an outlook on future activities.

## 2. MVP-L and MVP-E

The process modeling formalism MVP-L [BLRV95] was developed first at the University of Maryland and subsequently at the University of Kaiserslautern. MVP-L provides three basic elements: product models, process models, and resource models. Attributes can associated with each of these to enable data collection. The basic entities can be organized in aggregation and refinement hierarchies to form more complex models and to allow to view software process models on different levels of abstraction. Control flow is described by entry and exit criteria. Outstanding features of MVP-L are interfaces to measurement tools for automatic and manual data collection, and data aggregation specifications where the value of an attribute is computed by using values of attributes on a lower level of abstraction (e.g., “the status of a document is ‘complete’, if and only if all its parts have the status ‘complete’”). In its original version the language style was purely textual. When the language was used in industrial projects, it showed that a graphical representation schema was needed to facilitate understanding and communication among process performers [KNMS+92].

A number of tools has been developed around MVP-L. Together they are called MVP-Environment, for short MVP-E. In particular MVP-E comprises:

- MoST (Modeling Support Tool) provides a graphical user interface for functions to edit, check, and analyze textual process models [BV97].
- GEM (Graphical Editor for MVP-L) provides a user interface to MVP-L models in graphical notation, so that a process engineer can easily define process models or arrange a graphical layout of existing models for review by people involved in the development process [Ver97].
- MVPsim is a simulator for the purpose of risk assessment. Stochastic models (e.g., effort, failure rate) are used to implement quality models expressing attributes of entities. Monte Carlo simulation allows for the determination of overall distributions of several quality aspects [Brö96].
- The process engine MVP-S defines MVP-L’s operational semantic. A distinct feature are the mechanisms for on-line (i.e., during process performance) data collection MVP-S provides [Lot96]. MVP-S was realized to analyze the mechanism for interaction between user and process engine.
- The translator Pamela takes files containing MVP-L process models and transforms them into data which can be interpreted by the commercial workflow system ProcessWEAVER. The ability to change the execution semantics on a high level very easily results in an experimental process-sensitive software engineering environment [VM97].
- ProTail (Process Tailoring Tool) addresses the need to tailor process models to project-specific contexts and goals. The process engineer can pick values from a list of predefined parameters or define new ones. Rules are applied to modify the process model in order to make it suitable for the project [MSV97].

These components can be grouped into the four categories Modeling, Simulation, Enactment, and Tailoring. Figure 1 depicts MVP-E and its components. The requirements for the tool set were derived from industrial process modeling projects (e.g., [KNMS+92]). Its application in industrial projects demonstrated that the use of the tools reduces the effort for model creation, validation, modification, and analysis.

In the remainder of this paper we will discuss the use of MVP-E’s component in process modeling case studies. Since MVP-E embodies a lot of feedback from the application in industrial projects, and therefore has reached a certain stage of maturity, it is an example that might be worthwhile looking at when modeling processes or developing support for process engineers. Further validation of the tool MoST include modeling of standards and reference models.

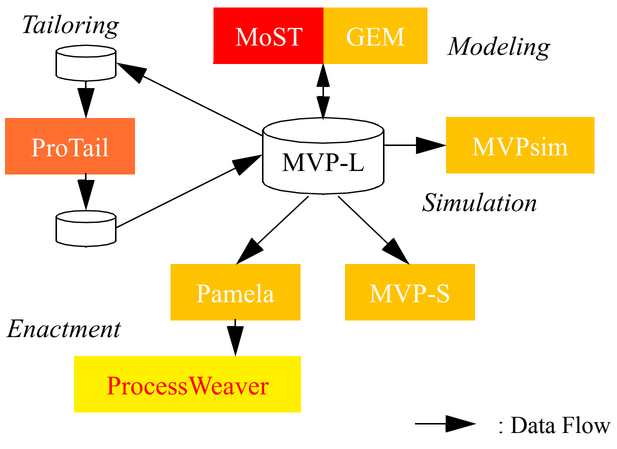

Figure 1: The Components of MVP-E

## 3. MoST

MVP-L's modeling support tool MoST [BV97] was developed at the University of Kaiserslautern in cooperation with the Fraunhofer Institute for Experimental Software Engineering as a tool to support process modeling. MoST supports a process engineer in the construction phases of a (descriptive) model written in MVP-L.

MoST provides three categories of functions: functions to facilitate the manipulation of process models (e.g., editing functions), functions to analyze the structural aspects of process models (such as product flow or hierarchies), and functions to check consistency (e.g., the check whether each model used has been defined).

The modular structure of MVP-L process models allows both their incremental development and further refinement of process models, as more information about the process to be modeled is available. Parts of process models can be deleted or reused from existing projects. The modification of single models is supported by text search functions. It is possible to jump directly to syntactically incorrect text lines of the model. MoST allows to save complete process models as well as single models.

The structural analysis functions examine product flow, i.e., which products are used in which processes and structural aspects of process and product hierarchies. A typical problem in this respect is that a product is defined, but never used within the process model. This indicates a possible flaw of the model or even a misunderstanding of the process.

Consistency rules check for semantic integrity of process models. This includes type consistency, consistency of refinement (i.e., if a process is refined: how are the products used distributed onto its subprocesses?), and checks for cycles in the process model. MoST provides two different classes of consistency rules: *intra consistency* rules check for consistency within single components, *inter consistency* rules check for consistency across different components within one process model. This allows the process engineer to work separately on single components before integrating them into a larger context. This strategy has been proven helpful especially when dealing with large process models: model fragments can be developed and checked separately before they are integrated into a larger process model.

## 4. Application of MoST and Lessons Learned

MoST is used for the description of descriptive and prescriptive process models. Larger examples of prescriptive models helped to validate the tool prior to its use for descriptive modeling in industrial studies. This section describes the modeling of those validation examples and discusses some of the lessons learned about the examples itself.

In order to create the process models presented in this section, the following strategy is used which has been defined consistently with various industrial projects (e.g., [BDT96], [BHV97]).:

First, all necessary products are identified and all elementary products are modeled. Thereafter the complete product hierarchy is built up in a bottom-up manner. Next, the processes have been modeled around the existing product models in the same way as the product model by starting with the elementary processes. This step also includes the specification of relations between process models and product models (product flow). In a last step some attributes are attached to the products and processes which are used to specify the control flow and conditions under which a process starts and terminates (entry and exit criteria).

### 4.1 Converting of Textual Representation into MVP-L

One application of MoST for validating the tool was modeling the book "Engineering Real Time Systems" by Rolv Bræk and Øystein Haugen [BH93], which is well-known in the distributed systems community. The book presents an integrated set of methods that gives a broad coverage of technical aspects of the development of complex real-time systems. The techniques offer support for the initial requirements specification, functional design, implementation design and for the later implementation. The Specification and Description Language SDL (defined by the CCITT) is used to express the functional design. The developed process model prescribes the strategy for engineering real-time systems with the SDL approach.

The main structure of the book was helpful in identifying the main process steps. Each part of the book could be modeled easily as a process step. The chapters covering the phases specification and SDL model creation are described detailed and therefore it was easy to refine the process steps and products. For subsequent process steps (e.g., design, or implementation), however, it was very difficult to identify refined the process steps and products. In some cases, products were refined across several refinement levels and some of these levels were never used by any other process step. Another important aspect was that the subsequent process steps could be either done automatically due to the usage of SDL or are not supported by SDL. The book tends to be rather an introduction into SDL with engineering real-time systems as an application of SDL than vice versa. Therefore, modeling the process steps with high usage of SDL was quite easy and modeling the other parts of the engineering process was difficult.

A lot of inconsistencies, incompleteness, or other ambiguous points in the process model were detected with the analysis and consistency functions of MoST. Some products have been refined much further than the corresponding processes which produce, consume or modify these products (e.g., the specification document has three refinement levels and the corresponding process steps only two). Some of the products have been produced by one process but have never been used (consumed or modified) by another process (e.g., nothing is said about when and how to use the produced test cases). Similarly, a few products were used by some processes even though they have never been produced by earlier process steps. Furthermore, some process steps are described vague with a number of alternatives (e.g., the reviews may be performed on several levels in a project and can be performed in many different ways).

Some other problems occurred with regard to the process modeling language: It is sometimes not necessary to define the creation time for refined products when using natural language. But when modeling the products, it must be determined at which point of time every product exists or not (e.g., the requirements are refined in several other products; for each of these refined products is stated when they are produced, but nothing is said about the upper level requirements document).

The usage of process modeling environments such as MVP-E enables to detect weaknesses and inconsistencies in textual representations as shown in this example. The quality of the textual representation will be improved and therefore, it will be easier for the reader to understand and apply the content of the book.

### 4.2 Modeling Standards

A second, more obvious, class of process models to be used prescriptively are standards and handbooks. MVP-L was used to formalize process standards [VM97], i.e. documents intended to describe processes in a high quality. The purpose was to assess whether MVP-L can handle large, abstract process descriptions. Case studies included the IEEE 1074-1991 standard [IEE94], the German Vorgehensmodell or short V-Modell [DBdI92], and a handbook describing Cleanroom processes [IBM91].

The transformation of structured but natural language descriptions requires an intensive review and interpretation of the standard. The advantages of using process modeling languages for documents to be used prescriptively are consistency, completeness, and unambiguity of the process model. First, formalization reveals inconsistencies in process documents (e. g., in the V-Model some attribute values do not match between different levels of process refinement which violates an MVP-L rule, or in the IEEE standard an activity, *Select or Develop Algorithms*, is marked as mandatory in the activity list of the design process and optional in the activity's description). Second, some aspects of completeness can already be checked easily by syntactical rules; other checks require more complex computations (e.g., completeness of refinement). Thirdly, ambiguity is detected by alternatives in formalizing the document.

It was found that the standards were to a high degree consistent, complete, and unambiguous. This might be the result of a number of reviews prior to disseminating the standards. However, when MoST was used to check an intermediate version of a German national standard the number of inconsistencies and incomplete parts was higher than expected.

The complexity of the processes was not easy to handle. Because standards are intentionally generic, some mechanisms must exist to express relationships among alternatives of the model that hold for specific contexts only (e.g., a rather simple example is that a database needs to be tested only if a database is implemented). It was found that in general it can be distinguished between three different approaches for handling generic process descriptions in existing standards or handbooks: a) provide a comprehensive process model and specify additional tailoring rules for deleting parts in the comprehensive model (e.g., V-Modell), b) provide a comprehensive process model where parts are specified as optional (e.g., IEEE standard 1074), or c) present focused process models that define only a particular type of process (e.g., STARS IR70E).

## 5. Additional Support for Process Engineers

The functionality of MVP-E extended with each new tool but there are still deficiencies. Most of the lacks were found in modeling larger processes. Especially the needs of process engineers should be regarded as highly relevant for process support environments. In the following some important aspects are highlighted which are only partially supported in MVP-E but should be improved from the viewpoint of a process engineer. They are subject of ongoing work. The mentioned suggestions can be considered as general recommendations or requirements for process support environments.

*Support reuse of process models*. Process models as described for example in standards or handbooks are usually of a generic type in the sense that they are only specified on an abstract level. What is usually missing are statements about the domain or context in which the models are valid as well as (empirically derived) guidelines on how to adapt the models to project-specific goals and characteristics. From the viewpoint of a process engineer a process support environment should provide mechanisms to cope with process variants, i.e., the tailoring of process models should be supported. Important research issues concern an understanding of relevant variation parameters, an appropriate representation of generic process knowledge, the development of tailoring mechanisms and the packaging of process models for reuse in further projects. It should be investigated, whether *internal* or *external* mechanisms are more useful for tailoring. "Internal" means that the process model itself contains genericity (e.g., optional product flow parameters) like the IEEE standard; "external" means that the process model is an input to an application (e.g., generator) which allows tailoring according to project characteristics and goals like the V-Model proposes. A related question is, which software reuse approach (e. g., templates, generation, composition, transformation) can be applied to software process models. The prototypical tailoring tool *ProTail* for MVP-L [MSV97] uses a transformational approach. First experiences with this tool show enormous time savings in constructing the context-specific model. Beyond this the transformational approach seems to be a good solution for managing the numerous interrelations between process fragments (e. g., control flow, product flow, refinement hierarchies).

*Support change management*. When the goals or characteristics of a project change, the processes must react accordingly. Consequently the process models, which should always reflect the real world situation, must be updated. This requires flexible mechanisms which allow for alternating modeling, planning and enaction. General problems exist manipulating states and keeping them consistent with the type information, and partial backtracking of process tracks in the case of erroneous process performance. The process engine MVP-S enables flexibility through formulating "weak" entry and exit-criteria. Backtracking or on-line refinement during enaction, which could be especially important for supporting "creative" process elements, is not yet provided.

*Support goal-oriented instrumentation of project plans*. From the viewpoint of a process engineer measurement especially helps in understanding software processes and the relationships between them as well as in improving processes over time. Basili states that measurement must be focused, based upon goals and models [Bas93]. Goal-oriented measurement requires a framework which comprises how to formulate goals, how to trace those goals to measures and how to interpret the gathered data. A possible approach for defining measurable goals is the Goal/Question/Metric Paradigm (GQM) [BW84]. MVP-E supports measurement through the attribute concept which allows for observing and controlling project states as well as giving feedback to all involved persons (e. g., developer, quality assurance, manager). What is needed is a systematic tool-based method for coupling measurable goals and project plans. This is currently done by manually generating measurement plans and linking the measures afterwards to corresponding MVP-models [BDT96]. Tool support should be available for the creation or generation of measurement plans, automatic consistency checks between measurement- and project plans, and finally the integration of both. Scientific issues comprise appropriate consistency rules and the formal representation of measurement plans.

*Support different representations*. MVP-E supports mainly three representations of MVP-models. The graphical and textual language representation are suited for the modeler. The enactment representation which shows a part of the project state supports the developers and the project manager. A larger set of possible representations is needed in order to increase the acceptance of a process support environment. For example, a quality assurance manager is mainly interested in the quality attributes, their values and the difference to the desired values. Experiences in modeling have shown that the modeling of only certain aspects of process models should be reflected in a suited representation. Process support environments should have the ability to define role- and task-specific representations of the models. Research issues comprise adequate presentations styles, the definition of sufficient information sets and mappings between different representations. [TM96]

*Support global process management*. New forms of interconnected global organizations (e. g., virtual cooperations [DM92], network organizations [SMC92], global learning organizations [MR94]) require support for global software processes. Such organizations are characterized by distributed resources (e. g., developers, knowledge, tools, budget), distributed authority (e. g., responsibilities, delegation mechanisms, formal rules), a large set of different development environments and non-hierarchical decision processes. Traditional management hierarchies are to be replaced by organizational structures which allow to distribute power according to who has the relevant resources, informations and capabilities to contribute to the task at hand. This leads to several consequences for process support environments. In contrast to local process support issues like global decision support, cooperation with time delays, or crossing cultural barriers arise. New delegation and decision processes are needed as well as the integration of different globally distributed development processes. Another challenging issue especially relevant in this context is the integration of CSCW and software process technology. From the viewpoint of a process engineer separate modeling of different views and their integration should be supported. Apart from implementational details the basic principles of MVP-E allow for global process enaction. Global modeling can be supported indirectly by separate modeling of different views on a process and their tool-based integration with MoST. An integrated conceptual framework for handling the "global" aspects of software processes should be investigated in future. First approaches concerning parallel descriptive modeling and distributed decision processes already exist [VDMM96, BW97].

## 6. Conclusion

Numerous systems have been built within the past decade focusing on many issues of software process technology [FW96]. The majority of them was designed for supporting the performance of a project by representing states and guiding developers. Examples range from flexible workflow management systems, and object-oriented database systems, to distributed open environments, or all of them. The research in this area has demonstrated a definitive level of immaturity which is due to the complexity of the goal itself. Major problems in the area of so-called process-sensitive software engineering environments must be solved before they are ready for use in real development projects [MP93]. Nevertheless, some success stories do exist in which technology developed in this particular research field has been transferred successfully to other domains such as workflow management systems (e.g. LEU from LION GmbH, Germany or Process WEAVER from Cap Gemini, France).

Abstract process representations are essential to allow for better communication about the processes, thus serving as a reference throughout measurement and change. However, there is a considerable lack of success stories about software process technology being used in industrial improvement programs. The reasons for this lack might be a mixture of the following:

- Software process technology is used in industry less widely than expected and consequently it still needs to be transferred into industrial practice.
- Software process technology is used "behind the curtain" which means that no value is seen in distributing messages about their usage in industrial projects.
- Software process technology is substituted by approaches from other domains (e.g., business process engineering) or just home made applications.
- Software process technology does not meet process engineers' needs because it was developed in purely academic environments, apart from industrial experience and people directly involved in the processes.
- Software process technology as a field is still in an immature state where powerful and widespread tools can be realized as soon as basic fundamentals are detected and manifested into components (e.g., frameworks for user agendas, or standard product database systems).

MVP-E can be seen as an example process modeling environment for building large industrial software development processes. The use of MVP-L in industrial projects has helped to build a system that supports the process engineer in improvement programs. We see MVP-E as considerably different to many other system builds in the area of software process technology because it does not focus on processes described on a fine-grain level (i.e., process implementations) but allows for the management of process models on a coarse-grain level (i.e., process design). What is remarkable about MVP-E is not a single feature solely but the aggregation of the following issues:

- MVP-E is a highly integrated environment which allows tools focusing on multiple aspects of software process technology to work together. From descriptive modeling to simulation and enactment many aspects of project support are covered. This tight integration demonstrates the feasibility of comprehensive support for process engineers.
- MVP-E was developed especially for software process modeling. The tools share some knowledge about these processes which makes them more powerful than tools adapted for that purpose.
- MVP-E was validated in industrial improvement programs. Although still a prototype it has demonstrated its suitability and usefulness for the process engineer. Incremental builds helped to recognize feedback from projects which drove the development in those parts of the system that were most promising. For example, a graphical interface was developed late in the project whereas checks that were hard to perform manually were automated first.
- MVP-E addresses the need for specifying measurable properties of software development processes and products. Language constructs allow for an adequate representation of relationships among levels of refinement. One can define control flow by using entry and exit criteria but the primary reason for these constructs is the definition of valid states for process performance.

## 7. Future Work

Two related projects (MILOS, SPEARMINT) exist, which originate from MVP-E and focus on specific topics mentioned in Section 5. These projects are based on the basic modeling concepts of MVP-L. They differ in detailed modeling concepts, enactment features and implementation details.

MILOS (Modeling Language and Operational Support for Software Processes) [VDMM96] expands the support of MVP-E to “creative” processes. Creative means both stochastic enactment (i.e., the plan must be highly generic) and the absence of knowledge about how to proceed in advance (i.e., there is no plan). Therefore AI/KE planning concepts are integrated. The MILOS-Project is conducted at the University of Kaiserslautern as part of the Sonderforschungsbereich (Special Research Project) 501. Its main goal is to provide a powerful process-sensitive software engineering environment which allows process evolution, i.e., replanning a project under enaction.

Experience gained especially with the modeling components MoST and GEM, will provide valuable input for a new software process modeling environment, SPEARMINT (Software Process Elicitation, Analysis, Review and Model Integration Tool), which is currently being developed at the Fraunhofer Institute for Experimental Software Engineering. One of SPEARMINT’s major characteristics will be its ability to provide different views of the process model, some of them in a graphical representation. The tool will therefore allow the process engineer to focus on different aspects of the process model, at the same time avoiding information overflow. This had been shown to be a major deficiency of MVP-E. Since SPEARMINT is intended to be a pure modeling tool with focus on measurement [BW97], it will also provide an interface to MVP-E, the main integration factor being the internal representation of the language MVP-L. This will enable simulation, enactment, and tailoring.

It is planned to integrate the different systems by developing interfaces. It is expected that herewith a larger competence and experience in the process modeling field can be integrated.

## 8. Acknowledgments

We would like to thank everybody who helped to realize MVP-E. Special thanks go to Christopher Lott for supervising the work on the process engine and to Alfred Bröckers for doing the simulator part. Both provided valuable feedback to the development of all the tools. All the students who participated in the MVP-E activities did a great job in implementing the system’s components.

## 9. References

[Bas93] Victor R. Basili. Applying the Goal/Question/Metric paradigm in the experience factory. In *Presented at the 10th Annual CSR Workshop, October 1993*, 1993. To appear as part of a book entitled “Software Quality Assurance: A Worldwide Perspective” to be published by Chapman and Hall.

[BDT96] Alfred Bröckers, Christiane Differding, and Günter Threin. The role of software process modeling in planning industrial measurement programs. In *Proceedings of the Third International Software Metrics Symposium*, Berlin, March 1996. IEEE Computer Society Press.

[BH93] R. Bræk and O. Haugen. *Engineering Real-time Systems: An object-oriented Methodology using SDL*. Prentice Hall, New York, London, 1993.

[BHV97] Ulrike Becker, Dirk Hamann, and Martin Verlage. Descriptive modeling of software processes. International Software Engineering Research Network (ISERN) Technical Report ISERN-97-10, Fraunhofer Institute for Experimental Software Engineering, 1997.

[BLRV95] Alfred Bröckers, Christopher M. Lott, H. Dieter Rombach, and Martin Verlage. MVP–L language report version 2. Technical Report 265/95, Department of Computer Science, University of Kaiserslautern, 67653 Kaiserslautern, Germany, 1995.

[Brö96] Alfred Bröckers. *Modellbasierte Analyse von Software-Projektrisiken*. PhD thesis, Department of Computer Science, University of Kaiserslautern, November 1996. Also published by Shaker Verlag, Aachen, Germany.

[BV97] Ulrike Becker and Martin Verlage. MVP–L’s modeling support tool MoST. Technical Report IESE-002.97/E, Fraunhofer Institute for Experimental Software Engineering, Sauerwiesen 6, 67661 Kaiserslautern, Germany, February 1997.

[BW84] Victor R. Basili and David M. Weiss. A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering*, SE-10(6):728–738, November 1984.

[BW97] Ulrike Becker and Richard Webby. 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.

[DBdI92] Der Bundesminister des Innern. Planung und Durchführung von IT-Vorhaben: Das Vorgehensmodell. Technical report, KBSt, Graurheindorfer Str. 198, D-5300 Bonn 1, August 1992.

[DM92] William H. Davidow and Michael S. Malone. *The virtual corporation: Structuring and revitalizing the cooperation for the 21st century*. Harper Business, New York, 1992.

[FW96] Alfonso Fuggetta and Alexander Wolf, editors. *Software Process*, volume 4 of *Trends in Software*. John Wiley & Sons, 1996.

[IBM91] IBM. Software Technology for adaptable, reliable Systems (STARS) program. Technical Report Task IR70E, IBM, 1991.

[IEE94] Institute of Electrical and Electronics Engineers. *IEEE Standards Collection – Software Engineering – 1994 Edition*, 1994.

[KNMS+92] C. D. Klingler, M. Neviaser, A. Marmor-Squires, C. M. Lott, and H. D. Rombach. A case study in process representation using MVP–L. In *Proceedings of the Seventh Annual Conference on Computer Assurance (COMPASS 92)*, pages 137–146, June 1992.

[Lot96] Christopher M. Lott. *Measurement-based feedback in a process-centered software engineering environment*. PhD thesis, Department of Computer Science, The University of Maryland, College Park, Maryland 20742, February 1996.

[MP93] Nazim H. Madhavji and Maria H. Penedo. Guest editor’s introduction. *IEEE Transactions on Software Engineering*, 19(12):1125–1127, December 1993. Special Section on the Evolution of Software Processes.

[MR94] M. J. Marquardt and A. Reynolds. *The global learning organization: Gaining competitive advantage through continuous learning*. Burr Ridge, 1994.

[MSV97] Jürgen Münch, Markus Schmitz, and Martin Verlage. Tailoring Large Process Models on the Basis of MVP-L. In Sergio Montenegro, Ralf Kneuper, and Günther Müller-Luschnat, editors, *Proceedings of the 4th Workshop der Fachgruppe 5.1.1 (GI): Vorgehensmodelle – Einführung, betrieblicher Einsatz, Werkzeug–Unterstützung und Migration*, Berlin, Germany, March 1997. In German.

[SMC92] C. C. Snow, R. E. Miles, and H. J. Coleman. Managing 21st century network organizations. *Organizational Dynamics*, 20(3), 1992.

[TM96] Josée Turgeon and Nazim M. Madhavji. A Systematic, View-Based Approach to Eliciting Process Models. In Carlo Montangero, editor, *Proceedings of the Fifth European Workshop on Software Process Technology*, Nancy, France, October 1996. Lecture Notes in Computer Science Nr. 1149, Springer–Verlag.

[VDMM96] Martin Verlage, Barbara Dellen, Frank Maurer, and Jürgen Münch. A synthesis of two process support approaches. In *Proceedings of the 8th Software Engineering and Knowledge Engineering Conference (SEKE'96)*, pages 59–68. Knowledge Systems Institute, Skokie (IL), USA, June 1996.

[Ver97] Martin Verlage. *Ein Ansatz zur Modellierung großer Software–Entwicklungsprozesse durch Integration unabhängig erfaßter rollenspezifischer Sichten*. PhD thesis, Department of Computer Science, University of Kaiserslautern, July 1997.

[VM97] Martin Verlage and Jürgen Münch. Formalizing software engineering standards. In *Proceedings of the Third International Symposium and Forum on Software Engineering Standards (ISESS '97)*, Walnut Creek, California, USA, March 1997. IEEE Computer Society Press.