The Fraunhofer IESE Series on Software and Systems Engineering

Jürgen Münch
Ove Armbrust
Martín Kowalczyk
Martín Soto

Software Process Definition and Management

Fraunhofer
IESE

Springer

The Fraunhofer IESE Series on Software and Systems Engineering

Jürgen Münch • Ove Armbrust •
Martin Kowalczyk • Martín Soto

Software Process Definition and Management

Springer

Jürgen Münch
University of Helsinki
Department of Computer Science
Helsinki
Finland

Ove Armbrust
Alpine Electronics Research of America
Torrance, CA
USA

Martin Kowalczyk
Fraunhofer IESE
Kaiserslautern
Germany

Martín Soto
eleven GmbH
Berlin
Germany

ISBN 978-3-642-24290-8 ISBN 978-3-642-24291-5 (eBook)
DOI 10.1007/978-3-642-24291-5
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012936487

ACM Codes: D.2, K.6

Foreword

One of the most significant contributions of the agile methods community has been to put to rest the mistaken belief that there could be a one-size-fits-all software process by which all software systems could be developed. The agilists not only produced a family of process models that were clearly different from the traditional single-pass, sequential, requirements-first models, and their attendant baggage, but also they provided evidence of their successful application, often in situations in which the traditional approaches had failed. Further, they were willing to admit that their methods were not a panacea for all projects. For example, agilist Kent Beck stated in his pioneering 1999 book, Extreme Programming Explained, that "Size clearly matters. You probably couldn't run an XP project with a hundred programmers. Not fifty. Nor twenty, probably. Ten is definitely doable." (As an example of the pace of process technology, an increasing number of organizations have successfully evolved Architected Agile processes using a combination of architecting, XP practices, and a Scrum-of-Scrums approach to scale up to about 100-person teams—but not further to date).

Once one accepts that multiple types of processes are going to be needed for different project situations, a whole new field of questions arises. What are the process driver factors that lead projects toward more agile, more plan-driven, more risk- and value-based, or other methods? How does the existence of large, cost-effective, but often incompatible COTS products or cloud services affect a project's processes? What sort of processes best fit a project that must provide high levels of confidentiality, integrity, and availability assurance while being rapidly adaptable to high rates of change? How do factors such as corporate or national cultures affect a project's choice of processes? How does a project cope with the need to integrate different process models being used in different parts of the project? How does an organization evolve from an opportunistic quick-to-market process as a startup, to a high-assurance process once the product has a large customer base to satisfy? How does an organization evaluate the maturity and domain of applicability of new process approaches?

The number, variety, and importance of such process questions have caused many organizations to appreciate the need for a much broader and adaptable approach to software processes, including standards groups, professional societies, the Software Engineering Institute, and some government organizations. But there is a large amount of inertia to overcome, in terms of traditional standards, guidelines, contracting mechanisms, entrenched bureaucracies, and course curricula. Thus, there is a great need for well-organized guidance about the properties and areas of strength and weakness of various classes of software and system development and life cycle processes.

This book provides a major step forward in providing such guidance. It is written by authors with a wide variety of experience in commercial, industrial, government, and entrepreneurial software processes. It provides an organized approach for addressing the questions above, and numerous other questions, by describing and distinguishing among various classes of process technology such as prescriptive and descriptive processes, process modeling and simulation languages and tools, experimental and observational process evaluation approaches, and process improvement approaches.

Following the Osterweil "Software processes are software too" insight about the duality between software products and processes, the book addresses software process counterparts to software product technologies such as software process requirements engineering, architecting, developing, evolving, execution control, validation and verification, and asset reusability. It provides good illustrative examples of their use, well-worked-out definitions of process terms, and question-and-answer assignments for use in teaching software process engineering to students or practitioners.

As a bottom line, this book has arrived at an opportune time to help many classes of software-reliant people and organizations learn how to cope with a multiparadigm software process world. These include software-reliant enterprise managers and their staffs; software-intensive project managers, systems engineers, and developers; academic faculty researchers and teachers; and a growing body of next-generation software process engineers. If you fit into any of these classes, I believe that you will benefit greatly from reading this book and having it around for future reference.

Los Angeles, CA, USA Barry Boehm

Preface

The concept of processes is at the heart of software and systems engineering. Software process models integrate software engineering methods and techniques and are the basis for managing large-scale software and IT projects. High product quality routinely results from high process quality. Process management deals with getting and maintaining control over processes and their evolution.

Who Should Read this Book?

This book is aimed at students in undergraduate and graduate courses, at practitioners who are interested in process definition and management for developing, maintaining, and operating software-intensive systems and services, and at researchers. Readers of this book should have basic familiarity with software development.

1. *Students.* The book can be used in general software engineering courses, in specialized process management courses, or in courses such as software project management, software quality management, software process improvement, or software measurement. The book may also be interesting for students who want to get a focused introduction to software process management, but would rather avoid general software engineering textbooks that typically present comprehensive process models with canned technology or nonintegrated development techniques.
2. *Practitioners such as project managers, process engineers, or consultants.* Practitioners may find the book useful as general reading in order to become familiar with the topic, for updating their knowledge, for understanding the relationships between process management and other aspects of their daily work, and for better assessing the relevance of the topic. Besides project managers, the book is especially relevant for process engineers, consultants, software engineers, SEPG members, members of process improvement groups,

heads of software development departments, quality managers, project planners, and coaches.

3. *Researchers*. Although the maturity of software process management practices in industry has increased and the state of software process research has advanced, the field is still quite immature. Students, practitioners, as well as researchers should be aware of the limitations of existing process management technologies, know the deficiencies of existing process models, and understand unsolved problems in the field. There is still a long road ahead toward mature software process management. We challenge software process researchers to address the vision that by using an appropriate combination of process and product engineering techniques, value creation for customers, adherence to cost and schedule constraints, and the fulfillment of quality requirements can be guaranteed on the basis of empirical facts.

Why a Textbook on Process Definition and Management?

One might argue that there are already many textbooks that include descriptions of software process models. The answer is "yes, but." Becoming acquainted with existing software process models is not enough. It is tremendously important to understand how to select, define, manage, deploy, evaluate, and systematically evolve software process models so that they appropriately address the problems, applications, and environments to which they are applied. Providing basic knowledge for these important tasks is the main goal of this textbook. There are many reasons that argue for a software process textbook:

Industry is in search of software process management capabilities. The emergence of new job profiles in the software domain (such as the agile coach, Scrum Master, process engineer, or offshore development coordinator), the lean and agile transformation of many organizations, and the establishment of so-called Software Engineering Process Groups emphasize the industry's need for employees with software-specific process management capabilities. Most of today's products and services are based to a significant degree on software and are the results of large-scale development programs. The success of such programs heavily depends on process management capabilities, because they typically require the coordination of hundreds or thousands of developers across different disciplines. Additionally, software and system development is usually distributed across different sites and time zones. To make things even more complex, technical and business environments as well as project goals often change during project execution, and an organization has to react to this in a controlled manner. The situation is similarly complex for operation and maintenance projects. Can such endeavors be mastered by using nothing but the appropriate software development and quality assurance techniques? The answer is "no, not at all." Analyses of large-scale development programs have shown that, with few exceptions, the reasons for the failure of such programs were not technical [1]. They almost always fail due to management problems. Due to the fact that process models glue together all activities, products, and resources, the relevance of process models for project management and especially for the success of large-scale software development programs is enormous. This need for process management capabilities is contrasted by the typical capabilities of young software engineering professionals. Such first-time employees are usually skillful with respect to software techniques such as coding or testing, but only have a marginal command of process management knowledge. This book provides basic building blocks of process management, such as process modeling or improvement, in order to lay a solid foundation for successful, sustainable processes.

Professional software engineers must fulfill process obligations. The duties of professional software engineers with respect to adherence to process models are becoming increasingly important. In order to illustrate this, let us compare a program that is developed by a student and a program that is developed by a software organization: Student programs usually solve small problems and are built to demonstrate that they work. If a student program fails, the consequences are limited. The student might not see the advantages of following a defined process because he does not coordinate his tasks with others and defects can be fixed without further consequences. If we consider the development of a software program by professional developers in a company, the situation is quite different: Each developer's personal work needs to be coordinated with the work of others; there is a customer paying for it, and the customer's business might depend on the resulting software. Thus, quality requirements are very important and the effects of potential failures are more serious or not tolerable at all. This means that there is often no way to avoid the definition, deployment, and control of high-quality development, operation, and maintenance processes. It is not sufficient anymore that a developer or a development team is convinced that specific quality requirements are fulfilled. Other parties such as customers also need to be convinced. Adherence to state-of-the-art processes and process management practices plays a crucial role when it comes to convincing others or even proving to them that quality requirements are fulfilled. This book explains different approaches to process improvement and conformance in order to support practitioners with respect to fulfilling process obligations.

Applicable knowledge from other disciplines is missing. Knowledge from other disciplines such as production engineering or business process management has only limited applicability for the software domain. One main reason is that production and business processes are typically repetitive processes in the sense that the same, well-understood process is enacted again and again with no or only minor variations. Quality assurance, for instance, is typically treated in production engineering and business management as the planning and deployment of a stable production or business process. Quality requirements can be fulfilled under given organizational constraints by just repeating this process. The situation is significantly different in software engineering: Software development is always the creation of an individual product. Therefore, process management cannot be based on the paradigm of repeatable processes. Quality cannot be achieved by just repeating processes. In the software domain, process and quality models need to be adaptable to individual development projects. There are no software development processes that fit for all types of projects or development environments. Consequently, approaches and techniques from production engineering or business process management (such as Statistical Process Control) cannot be transferred without difficulties. Sometimes they are useful when adapted appropriately. People who only have a production engineering or economic science background lack important capabilities for managing software projects and software processes. Software-specific process management capabilities are needed. This book introduces proven software-specific approaches to process management in order to support software engineers in their projects.

How Is the Book Organized?

Process management can be roughly divided into three areas: activities, infrastructure, and models (Fig. 1). Process management activities (left column in Fig. 1) can be seen as central: They consume, create, or modify different kinds of models (bottom part of Fig. 1), and are supported by a process management infrastructure (right column of Fig. 1).

The rationale for structuring the book is as follows (see middle part of Fig. 1): The basic concepts are given at the beginning. Afterward, existing representative process models are presented in order to give the reader an idea of what kinds of models exist and what they look like. A description of how to create individual models follows, and the necessary means for creating models (i.e., notations and tools) are described. Finally, different possible usage scenarios for process management are given (i.e., process improvement, empirical studies, and software process simulation).

Many books present practices, individual process models, or process standards in rich detail. However, there is often no description of how to customize these process models to a specific environment in a systematic way, information about the effects in specific project environments is not provided, and underlying assumptions are not true for many real situations (e.g., the assumption that development is performed in a colocated manner). As a consequence, project managers do not learn enough to assess the suitability of the presented models with respect to their own project goals and environments. Process engineers do not learn enough about how to customize or design appropriate models. In this textbook, we aim at providing knowledge that enables readers to develop useful process models that are suitable for their own purposes. In other words, the emphasis is not on working with given process models but on developing useful process models. Therefore, this textbook includes aspects such as descriptive modeling, continuous improvement, empirical studies, simulation, and measurement.

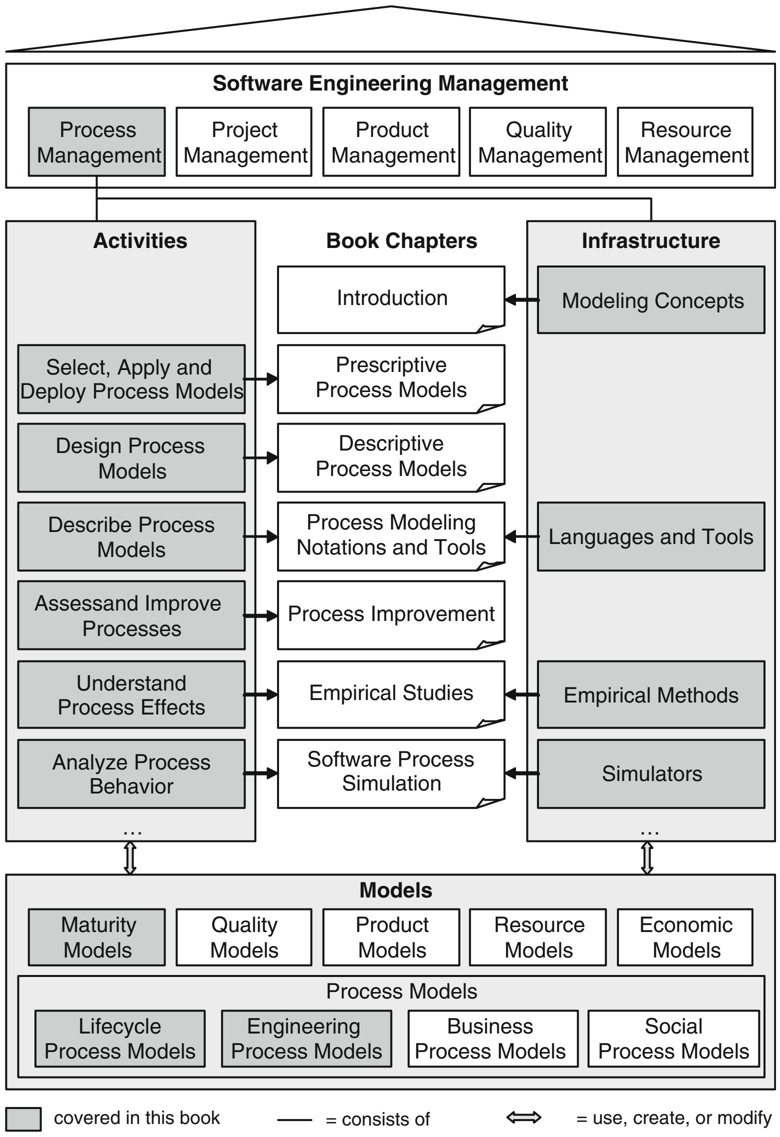

Fig. 1 Structuring of software process management

Reading. Although the chapters are self-contained, we recommend reading the book in a sequential order. Each book chapter starts with a short summary and a description of the chapter objectives. For each chapter, literature references, associated exercises, and sample solutions are given. The exercises aim at repeating and refining the material. They help the reader to get a better understanding and think about the contents from different perspectives. In addition, a glossary and an index are given so that the book can also be used as a reference book. The chapters focus on the following topics:

Introduction. In this chapter, the need for software process management and process models is motivated and the basic concepts and terminology are presented.

Prescriptive Process Models. In this chapter, prescriptive software process models are classified, a number of widely used process standards is introduced, two types of process representations are introduced (process handbooks and electronic process guides), and an exemplary deployment strategy is described. This helps to get an overview of existing models, to understand their advantages and disadvantages, and to get an understanding of what it means to select and deploy a process model.

Descriptive Process Models. In this chapter, a method is described for designing a process model based on observing current practices in an organization. Due to the fact that software engineers only change their behavior in small increments, the design of process models should start with or incorporate current practices of an organization. Deploying a process model that is too distant from currently lived practices implies high risks of nonacceptance. In addition, it is immensely important that the documented processes in an organization reflect the current practices. Quoting Watts Humphrey, "if you don't know where you are, a map won't help"—meaning that improving processes efficiently requires an understanding of the current practices. Therefore, this chapter puts a focus on descriptive process modeling for creating process models that match their counterparts in reality.

Process Modeling Notations and Tools. In this chapter, a characterization scheme for process modeling notations is given and selected notations are presented. One of the aims of this chapter is to show that different notations serve specific purposes differently, and that it is necessary to carefully consider which notation to choose. The so-called multi-view process modeling language, MVP-L, is described in more detail as an example of a notation that provides comprehensive modeling concepts. In addition, a reference framework for a process engineering tool infrastructure and an example tool are presented.

Process Improvement. In this chapter, different types of process improvement and assessment frameworks are presented, especially continuous and model-based approaches. In addition, selected software measurement and business alignment approaches are presented due to their significant role for process improvement.

Empirical Studies. In this chapter, a brief overview is given on how to determine the effects of a process model in a concrete environment. Such effects can be, for instance, the reliability of a developed code module, the defect detection rate of an inspection process, or the effort distribution of a life cycle process model. Software processes are, to a large extent, human-based and consequently non-deterministic. In addition, they are heavily context-dependent, i.e., their effects vary with the development environment. Therefore, empirical studies of different types are needed to understand and determine the effects of processes and to analyze risks when changing processes or introducing new ones.

Process Simulation. In this chapter, process simulation is introduced as a means for analyzing process dynamics. It is shown how simulation models can be created and how they can be combined with empirical studies to accelerate process understanding and improvement. In addition, a library of existing model components ready for reuse is introduced.

What Are the Benefits for the Reader?

Readers will gain knowledge and skills for designing, creating, analyzing, and applying software and systems development processes. In particular, the essential learning objectives of the book are:

– Understanding the importance of software processes and software process improvement
– Becoming acquainted with industrial software and system development processes and process standards
– Understanding the advantages and disadvantages of different process management techniques and process modeling notations
– Getting basic knowledge for modeling and analyzing software and system development processes
– Being aware of process management activities in software-related organizations

After studying the book's contents, readers will be able to contribute to process management activities, especially to applying common methods and notations for process modeling, designing software development processes, defining process improvement goals, selecting software process improvement approaches, participating in improvement programs, increasing process maturity, assessing processes, and evaluating processes by performing empirical studies.

Who Are the Authors?

The foci of this book were selected based on the comprehensive process management and software engineering experience of the authors. Although this book is intended as a general introduction to software process definition and management, it places an emphasis on specific areas, whereas others might highlight different aspects. The authors of this book have defined many organizational process standards, were involved in a multitude of industrial software process improvement programs, and have conducted many empirical studies of different types. They have produced national and international process standards for organizations such as the European Space Agency (ESA), the Japan Aerospace Exploration Agency (JAXA), and other governmental authorities. Some of the authors defined process assessment models and acted as certified process assessors for different schemes such as ISO/IEC 15504. Research-wise, the authors have developed, deployed, and evaluated a multitude of process management methods, techniques, and tools, including technologies for multi-view process modeling, process scoping, process tailoring, process compliance management, process visualization, and process evolution. All authors can draw on several years of experience as members of a process management division at a leading institute for applied research and technology transfer. They have helped many companies worldwide to improve their software development processes and their software process management. The authors also have significant experience in teaching software process management, be it by giving lectures at a university, in-house tutorials, or public seminars. The material presented in this book has been used, for instance, many times in a graduate process modeling course at the University of Kaiserslautern, Germany, and in a course of an accredited international distance education master program. One of the authors held several management positions in the area of process management, including being the head of a process engineering and technology group, the head of a process and measurement department, and the head of a process management division. In addition, the authors have served the scientific community in several ways such as co-organizing and contributing to the International Conference on Software and System Process (ICSSP), the International Conference on Product Focused Software Development and Process Improvement (PROFES), and the International Symposium on Empirical Software Engineering and Measurement (ESEM).

Helsinki, Finland	Jürgen Münch
Torrance, CA, USA	Ove Armbrust
Kaiserslautern, Germany	Martin Kowalczyk
Berlin, Germany	Martín Soto

Reference

[1] Humphrey WS, Konrad MD, Over JW, Peterson WC (2007) Future directions in process improvement. Crosstalk J 20(2):17–22