Software Processes and Life Cycle Models

Ralf Kneuper

Software Processes and Life Cycle Models

An Introduction to Modelling, Using and Managing Agile, Plan-Driven and Hybrid Processes

Springer

Ralf Kneuper
Dr. Ralf Kneuper Consulting
Darmstadt, Germany

ISBN 978-3-319-98844-3 ISBN 978-3-319-98845-0 (eBook)
https://doi.org/10.1007/978-3-319-98845-0

Library of Congress Control Number: 2018953100

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Foreword

A number of books have been written about software process improvement over the years, some of them quite good, some less so. Ralf has written a book that I will be happy to add to my library. Ralf has been involved with software process assessment and improvement for many years, so he has the experience necessary to speak knowledgeably about the topic. He is also addressing two topics that I think are important for systematic process improvement in today's world.

First, he is addressing agile methods. I became involved with agile nearly 20 years ago: I was asked to write a book chapter on Extreme Programming from the perspective of the Capability Maturity Model. I was impressed, surprised, and intrigued by the ideas captured in XP. While I would not agree with everything argued by the XP advocates, for the most part I found the XP practices appealing. I followed up with other agile methods, eventually becoming a Certified ScrumMaster. In my encounters with the agile community, I found a variety of perspectives, ranging from the "responsible center" to "fringe zealots".

I believe that agile methods have a great deal to offer the process community ... although there are those in both communities who downplay the contributions of the other. Process frameworks, such as CMMI, do not address every organizational need.

Software process as captured in the Software CMM, and now CMM Integration, focuses on building the capability of the organization to build systems. The emphasis is on operational excellence—meeting commitments, operating in an effective and efficient manner. There are other priorities that an organization could choose over operational excellence, such as innovation. In the custom software development world, operational excellence is crucial—but innovation cannot be ignored. In commercial software development, innovation may be the more important priority, yet meeting commitments is also useful.

Agile methods are focused on the needs of the software team to build a specific product in a specific context. As the agile experts all admit, you have to tailor the agile method to the unique needs of the project. If you tailor it too far, it may no longer be agile—but still be appropriate for the project's context. There are many good engineering and management practices embedded in the agile methods that

can and should be adopted by the software community, even when "agility" is not a major driver for the project.

Nearly 20 years ago, I listened to Bob Martin tell a story at XP Universe about someone he ran into in the hall. That person thanked Bob, telling him that his company had adopted XP and were delighted with it. Bob asked what he thought about pair programming—we don't do pair programming … well, what about the planning game—we don't do the planning game … how's continuous integration working out for you—we don't do continuous integration. Bob then asked, well what do you do? And the answer: we don't document anything! Sitting in the audience I felt a strong sense of schadenfreude (joy in the misfortune of others). More than once I've had someone tell me, we're doing something stupid! Why? The CMM told us to! In following up, it was never something the CMM actually directed, it was what people felt they needed to do to check a box.

People in both the process and agile communities need to approach "the way we build software" with both humility and a sense of inquiry. How can we do a better job of building software? Frameworks such as the Software CMM and now CMMI have many good ideas for the organization that can help in deploying effective methods, such as Scrum, Extreme Programming, and Feature Driven Development. I hope that Ralf's insights into the good ideas ranging from plan-driven to agile will help software professionals come up with good answers to the questions that they ask.

Second, Ralf is addressing the software process from the perspective of the Software Engineering Body of Knowledge. SWEBOK is an attempt to capture the critical ideas fundamental to good software engineering. The IEEE Computer Society solicited inputs from the software engineering community in a transparent, open, consensus process on what we know about building software effectively and efficiently. Integrating these insights into your software process improvement initiatives should be useful and important.

There are many other good sources of insight into software engineering and management that could be cited, but these are two of the most influential and widely known. I will not claim to agree with everything said in the agile community or by the IEEE Computer Society—I'm well known not to agree with many in the process world! Even when we disagree, keeping an open mind to potential insights and integrating those into our thinking, even when it causes us to change our minds, is the highway to continual improvement. Basing our decisions on empirical evidence as to what really works is the core high maturity at levels 4 and 5, but that's a different book.

Dallas, Texas, August 2017 *Mark Paulk*

Preface

Since software is of growing importance in today's world, the importance of the processes used to create, maintain and run such software is growing in parallel. These processes may be defined in detail or not at all, but even a software development project or organization that does not explicitly define their software engineering processes will use processes implicitly to do their work.

However, an increasing proportion of software engineering organisations defines their "set of interrelated or interacting activities which transforms inputs into outputs" which is the ISO 9000 definition of processes. Such process definitions may take very different forms, from very traditional, plan-based approaches with lengthy and sequential phases for analysis, design, implementation and test to agile approaches where essentially the same activities are performed within very short and frequent cycles, today usually called sprints. Experience over the last decades has shown that the best way to reliably get excellent products is to ensure that excellent processes are used to create and maintain these products.

Software engineering as a discipline integrates a variety of tasks ranging from "classical" computer science to business-related topics, and can roughly be distinguished into *software technology* that addresses the methodical and technological aspects, and *organisation and management of software development*, which, among other tasks, includes different planning, measurement and controlling tasks. Good software processes help to bring these different tasks together, working towards the common goal of delivering high-quality products on time and in budget.

As software becomes a vital component of many systems and affects many aspects of our day-to-day life, the processes for creating, maintaining and running software become critical as well. To efficiently develop high quality software that addresses their needs, companies have to align software development and software management with their business goals and processes. To achieve this, certain guidelines are needed for development, defining the activities, results and roles needed in more or less detail. This is particularly obvious in the context of regulated and safety-critical environments, such as automotive, medical devices, or financial services, but in different, e.g. more agile environments, alignment of the development

processes is similarly important, with the processes possibly taking a very different form, for example focussing on time to market rather than reliability.

Over time, plan-driven development and agile development have been developed as the two basic philosophies for software development. They both address the same objective of creating high-quality software in time and on budget, but with different emphasis. While agile development puts the emphasis on customer satisfaction and user benefit and tries to achieve these by making it easy to adapt work, plan-driven development puts more emphasis on requirements that have been identified in advance in order to achieve correctness and predictability. This results in different strengths and weaknesses of both philosophies, and in practice many development organisations use some combination of both.

Book Goals

Since process engineers and project managers face a diversity of approaches and standards that is hard to manage, defining and enacting appropriate processes constitute a challenging task that is often left to expertise and experience.

The book at hand therefore does not attempt to promote any specific approach, type of process, or model. Instead, it aims at delivering a big picture of the comprehensive field of software processes, covering in particular the essential topics:

- software process modelling
- software process models and life cycle models
- software process management, deployment and governance
- software process improvement (including assessment and measurement)

Furthermore, it can be used as a reference on software process models and notations, providing at least a brief overview of all the main approaches. The book also discusses the fundamental process principles, and presents an overview of current "hot topics" and emerging trends.

In particular, the book addresses the topics described in Chapter 8 "Software Engineering Processes" of the Software Engineering Body of Knowledge (SWEBOK® v3). To do so, it uses a uniform conceptual and terminological framework to present the software processes, explains the different topics by example, and shares experiences gathered over the years. The present book does not propose any new processes or methods. Its goal is to introduce software engineers into the topic of software processes to support the systematic development of high quality software in different and changing environments.

In discussing software processes and life cycle models, the focus lies on the benefits for development organisations and their customers that can be achieved, and what needs to be done in order to achieve them.

Conformance to relevant international standards is of course important as well, but should not be considered as the primary goal. Experience shows that most standards can be applied in very different ways, and a focus on compliance tends to lead

to inefficient processes that are therefore difficult to deploy in the organisation. A focus on the benefits of the processes, on the other hand, using standards as a tool to where useful or required, will in general be far more helpful and, as a result, also easier to deploy.

Target Audience

This book is aimed at graduate students, researchers, and professionals. It can be used as a textbook for courses and lectures, for self-study, and as a reference. When used as a textbook, it may support courses and lectures devoted to software processes, but also as complementing literature for more basic courses, such as introductory courses on software engineering or project management.

Software engineering processes provide a structure and guidance to the software engineering activities, and it will be difficult to understand the structure without an understanding of the basic activities that are structured by processes. To make effective and efficient use of the different methods and techniques of software engineering, appropriate processes are needed. This book helps to understand and use these processes, to identify which processes are most appropriate in a certain environment, to document them and to introduce them into the organisation or project.

However, the book at hand does not try to be a general software engineering compendium or textbook. Readers should already have some fundamental knowledge about computer science, software engineering, and project management. Specific tasks, such as software development, architecture, and quality assurance, are not part of this book, and readers are expected to bring in basic knowledge of these topics. Furthermore, basic knowledge about software economics is beneficial.

Outline

The overall structure of the book can be found in Fig. 0.1. In Chap. 1, the foundations of the topic are introduced, covering the basic concepts, a historical overview, and an introduction to the terminology used.

Next, Chap. 2 covers the various approaches to modelling software processes and life cycle models, before Chap. 3 discusses the contents of these models, addressing plan-driven, agile and hybrid approaches.

The following chapters address different aspects of using software processes and life cycle models in an organisation, looking at the management of these processes (Chap. 4), their assessment and improvement (Chap. 5) and the measurement of both software and software processes (Chap. 6).

Working with software processes is usually supported by different kinds of tools, which is the topic covered in Chap. 7, before a look at current trends in software processes in Chap. 8 concludes the book.

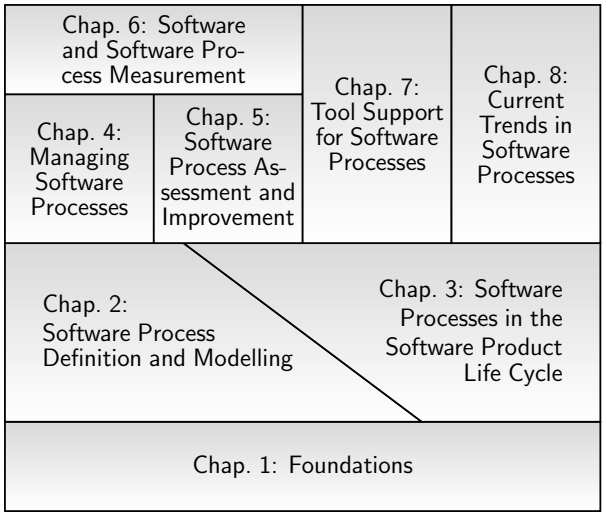

Fig. 0.1 Book outline

Relationship to SWEBOK®. The book is aligned with the *Software Engineering Body of Knowledge* (SWEBOK®) version 3, chapter 8 "Software Engineering Process" and, thus, provides a general introduction to the software process. SWEBOK® was created by the IEEE Computer Society, and its current version v.3 was published both as an IEEE guide and as ISO/IEC TR 19759:2015.

To some extent, the present book goes beyond the SWEBOK® contents by also providing insights into the topics of process selection and tailoring and by discussing emerging trends in the field of software processes. Furthermore, agile approaches are covered as well as plan-driven approaches, while SWEBOK has a strong emphasis on plan-driven development only.

Table 0.1 gives an overview of the top-level structure of the book and its relationship to SWEBOK®, Chap. 8.

Table 0.1 Relation of the present book to the topics covered by SWEBOK®, Chap. 8

SWEBOK® v3, Chap. 8	**This book**
Software Process Definition	Chapter 2, Chapter 4
Software Life Cycles	Chapter 3, Chapter 4
Software Process Assessment and Improvement	Chapter 5
Software Measurement	Chapter 6
Software Engineering Process Tools	Chapter 7

While Chap. 8 of SWEBOK® addresses software engineering processes as a topic of their own, several other chapters of the SWEBOK® also refer to software

engineering processes. Table 0.2 lists the main relevant parts of SWEBOK® and where these topics are covered in the present book.

Table 0.2 Relation of the present book to the topics covered by SWEBOK®, Chap. 7 and 10

SWEBOK® v3	This book
Chap. 7, Sect. 2.1 *Process Planning*	Sect. 4.4
Chap. 7, Sect. 3.4 *Monitor Process*	Sect. 4.7, Sect. 6.5
Chap. 10, Sect. 1.3 *Models and Quality Characteristics*	Sect. 5.2
Chap. 10, Sect. 1.4 *Software Quality Improvement*	Sect. 5.4

Just like software engineers should think about what happens to their software after development is completed, so this book goes beyond development processes and looks at the entire software life cycle including operations, service management and maintenance, as well as the governance of software processes.

Terminology. Where possible, the terminology used is usually based on the current (in 2017) version of the *Software and Systems Engineering Vocabulary* (SEVOCAB) which collects definitions of terminology from various other norms and standards published by ISO, IEC, IEEE and PMI, and expands this collection by a number of additional definitions. SEVOCAB is publicly available from `https://pascal.computer.org`, and also published periodically as ISO/IEC/IEEE 24765.

In particular, the terminology from SEVOCAB is usually used in the definitions in this book. Where this standards includes several different definitions of the same term, this book usually selects that definition that is most appropriate in the context of software processes and life cycle models.

On the other hand, the same definition is sometimes used in several different standards. In this case, only the sources most relevant in this context are given, always including SEVOCAB if the definition is included in this standard.

Case studies and examples. To help to get a good understanding of the topics discussed, examples and case studies are provided. In particular, the following two companies will be used for the case studies:

Case Study 0.1. (CS AutoSystems) *CS AutoSystems* develops and produces various electronic control units (ECU) for cars, e.g. electric window lifters. These ECUs consist of hardware as well as software, and in some cases they are safety-relevant, leading to high demands on their reliability as well as on the validation of the systems.

The development department of CS AutoSystems is fairly small, with about ten developers (hard- and software).

Case Study 0.2. (CS Insurance InfoSys) *CS Insurance InfoSys* is a large IT service provider, with about 1000 developers. The organisation develops and runs the information systems for its parent, a large insurance company. To a minor extent, it also acquires software from external suppliers which is then run in the data centre of CS Insurance InfoSys.

Both case studies are based on a combination of real, existing companies even though the companies as described do not exist in this form. Nevertheless, everything described in the case studies has actually happened in existing companies.

Example 0.1. Apart from these two case studies companies that will be used in case studies repeatedly across the entire book, various other examples will be used to illustrate the concepts introduced.

At the end of each chapter, references to further reading are included for those who want to go into more detail of one of the topics covered.

Also at the end of most chapters, some exercises are included to help get a better understanding of the concepts covered. These exercises do not just ask to repeat any contents described in the book but refer to applying and interpreting this contents in a certain context, sometimes the reader's own work environment. As a result, these exercises have rarely, if ever, a unique correct answer.

There are many norms and standards relevant in the field of software processes and this book addresses many of them. A list of the most important such norms and standards can be found in the appendix.

Acknowledgements. Many thanks go to Marian Benner-Wickner, Ernest Wallmüller, Eckhard Wirth and the anonymous reviewers for their feedback on various drafts of this book.

Many thanks also go to the copyright owners of the figures included. Unfortunately, not all copyright owners managed to answer this request, and I therefore had to remove a few figures I would have liked to include.

The author thanks the International Electrotechnical Commission (IEC) for permission to reproduce Information from its International Standard IEC 61508-3:2010. All such extracts are copyright of IEC, Geneva, Switzerland. All rights reserved. Further information on the IEC is available from http://www.iec.ch. IEC has no responsibility for the placement and context in which the extracts and contents are reproduced by the author, nor is IEC in any way responsible for the other content or accuracy therein.

Trademarks.

- ITIL® is a (registered) Trade Mark of AXELOS Limited. All rights reserved.
- PRINCE® is a (registered) Trade Mark of AXELOS Limited. All rights reserved.
- PRINCE2 Agile® is a (registered) Trade Mark of AXELOS Limited. All rights reserved.
- Capability Maturity Model®, Carnegie Mellon® and CMM® are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.
- CMMI and SCAMPI are registered marks of CMMI Institute LLC.
- Team Software Process, TSP, Personal Software Process, PSP and IDEAL are service marks of Carnegie Mellon University.
- PMI® and PMP® are registered marks of Project Management Institute, Inc.
- V-Modell® ist eine geschützte Marke der Bundesrepublik Deutschland. (V-Modell® is a registered mark of the Federal Republic of Germany.)
- SAFe® and Scaled Agile Framework® are registered trademarks of Scaled Agile, Inc.
- The Open Group® and TOGAF® are registered trademarks of The Open Group.
- IBM® is a registered trademark of International Business Machines Corporation.
- CORBA®, Object Management®, OMG®, and UML® are registered trademarks and BPMN™, Business Process Modeling Notation™, and Unified Modeling Language™are trademarks of the Object Management Group.
- COBIT® is a registered trademark of the Information Systems Audit and Control Association and the IT Governance Institute.
- IEEE® iis a registered trademark of the Institute of Electrical and Electronics Engineers, Inc.
- Microsoft® is a registered trademark of Microsoft Corporation.

About the author. Ralf Kneuper got a diploma in mathematics from the Univ. of Bonn, Germany, in 1985, and a Ph.D. in Computer Science from the Univ. of Manchester, UK, in 1989. Since then, he has worked with various companies on software quality assurance, quality management and software processes. Currently, he works both as an independent consultant on software quality management, process improvement and data protection, and as a professor of Business Informatics and Computer Science at the IUBH Internationale Hochschule in Germany. He has published extensively on CMMI, process improvement and process quality.

Contents