

## PREFACE

Silvia T. Acuña<sup>1</sup> and María I. Sánchez-Segura<sup>2</sup>

<sup>1</sup>*Departamento de Ingeniería Informática, Escuela Politécnica Superior,  
Universidad Autónoma de Madrid  
Avenida Tomás y Valiente 11, 28049 Madrid, Spain  
E-mail: silvia.acunna@uam.es*

<sup>2</sup>*Departamento de Informática, Universidad Carlos III de Madrid  
Avenida de la Universidad 30, 28911 Leganés, Madrid, Spain  
E-mail: misanche@inf.uc3m.es*

The software engineering discipline emerged in the 1960s as a consequence of the need to formalize the way software systems were developed. Since then a lot of research and development effort has gone into improving what have come to be termed software process models. The software process is a set of activities undertaken to manage, develop and maintain software systems, including the techniques to perform the tasks, the actors who execute the activities, their roles, constraints and the artifacts produced. Software process models are an abstraction of the software process which provides some useful information or insight for understanding, decision making and for enacting the actual process. Research in the 1990s was concerned with a variety of: not only prescriptive but also predictive and understanding-oriented<sup>4</sup> models. This was the consequence of a deeper understanding of the software process and the widening of the concerns of researchers who wished to investigate the impact of various organizational, social and economic factors on the software process.

One of the justifications for researching the software process is the view that the quality of the process has an impact on the quality of the

software, and that process improvement positively influences the organization's performance. There are at least two reasons for building, evaluating and using process models<sup>1</sup>:

1. To achieve better ways of defining and guiding development, maintenance and evolution processes.
2. To achieve better ways of improving processes at the level of individual activities and the process as a whole.

Ever since the earliest days of software process research, the above two motivations have been at the heart of investigation carried out in this area<sup>3</sup>. There has, therefore, been significant progress in the above two directions, and hence they are not the focus of this book.

Software process modeling has recently been dealing increasingly with new challenges raised by the tests that the software industry has to stand such as, for example, the need to produce applications at Internet-time, pressures for budget cuts and customers who are demanding more complex software that is easier to use. This book is intended to help in the dissemination and understanding of new software process model trends. The new trends covered in this book are related to:

- Processes for open source software
- Software process simulation for process improvement, management and decision-support
- Peopleware<sup>2</sup>, that is, the importance of people in the software process.

In other words, this book is intended to help readers understand the new software process models that are being developed to successfully manage new software development trends.

This book is structured as follows. The opening chapter explains an experience of implementing a process model for open source software. This is followed by three chapters (chapters 2, 3 and 4) that present the concept of the system dynamics approach to software processes improvement. Chapter 5 focuses on the new concept of people-oriented processes and what tools are available to support the enactment of these processes. Finally, chapter 6 recalls experience from describing the process model called E3 and the software system that supports this process model.

The discovery and building of process models for addressing new software development trends is known to be a long and costly process. Even so technological progress and the changing demands of today's society mean that the discovery and construction of new process models are always hot topics of research. One such new software development trend is the development of open source software. As such projects are a recent phenomena, the process models describing this type of development are not well known. The purpose of *chapter 1* then is to present a set of techniques for discovering, modeling, analyzing and simulating software development processes carried out in large open source software development projects based on public information accessible over the Web. Additionally, as an example of their applicability, the results of applying the defined techniques to a project with the above-mentioned characteristics, called NetBeans, are presented.

Simulation and dynamic modeling have been widely used as process improvement tools in industry. In recent years, this trend has also spread to the field of software process improvement. Hence, chapters 2, 3 and 4 focus on the description of work related to the use of simulation and dynamic modeling techniques in software processes.

*Chapter 2* presents a process framework that combines traditional techniques with process modeling and simulation techniques that support a qualitative and quantitative evaluation of the development process. This evaluation can be used to improve the software process, and is also a decision-making aid. The use of this framework can help software organizations achieve a higher process capability following to SEI's CMM (Capability Maturity Model)<sup>5</sup>.

*Chapter 3* includes a survey of the main process simulation applications since the 1990s. Additionally, this chapter describes IMMoS (Integrated Measurement Modeling and Simulation)<sup>6</sup>, a method for developing goal-oriented dynamic simulation models. By way of an illustration of the applicability of the IMMoS model, several cases of software process simulation models that were developed to support learning and decision making in software organizations within the automobile manufacturing industry are described.

*Chapter 4* presents an approach based on high level modeling for software projects. This approach separates the description of a particular project from the knowledge embedded in a software project model. The aim is to make useful complex system dynamics-based models that are built and adapted not only by experts but also by practitioners. Along these lines, this chapter describes a modeling and simulation process for system dynamics that allows the development of domain models and their specialization for particular problems.

*Chapter 5* addresses software development by means of people-oriented process models. These models have turned out to be very beneficial because they improve the quality of the interaction between people and processes. The chapter is divided into three parts focusing on the capture, visualization and use of the information by the people involved in the software development process. With respect to information capture, this chapter describes different knowledge process types and discusses the application of the GQM (Goal Question Metric) paradigm for data collection and/or to measure the particular process for which the data are captured. As regards the part of the process model related to the visualization of the information needed by each developer involved in a particular process, the generation of documents, role-based workspaces and control centers for software development are discussed. The use of the captured information is another important issue and is illustrated by discussing aspects concerning the management of previous experiences to assure that each experience can improve future development processes.

*Chapter 6* provides input for readers interested in learning about the evolution of process models. This chapter examines the evolution of an existing process model (E3) and the software system that supports this model, called the E3 system. E3 is a process model conceived to provide help for process/project managers, who construct and maintain models, and for practitioners, who use software process models. The chapter is a *post-mortem* analysis of the decisions that led to the E3 system requirements definition and gives insight into what principles any process model should follow if it is to remain useful years after being initially conceived.

## Acknowledgments

We would like to thank all the authors of the submitted chapters whose research has made this edited book possible. Particular thanks are due to Rachel Elliott who assembled the material and ensured that the presentation was consistent. We are especially grateful to Natalia Juristo and Juan F. Ramil for comments about the preface. We are also deeply indebted to Jens Heidrich, Chris Jensen, Cláudia Maria Lima Werner, Jürgen Münch, Márcio De Oliveira Barros, Rodion M. Podorozhny, Isabel Ramos, Günther Ruhe, Mercedes Ruiz and Guilherme Horta Travassos for helping us to improve the chapters of this book.

## References

1. Acuña, S.T. and N. Juristo. *Software Process Modeling*, International Series in Software Engineering, Vol. 10, Springer, NY, 2005.
2. DeMarco, T. and T. Lister. *Peopleware: Productive Projects and Teams*, 2<sup>nd</sup> ed. Dorset House, NY, 1999.
3. Derniame, J.C., B.A. Kaba and D. Wastell (Eds.). *Software Process: Principles, Methodology and Technology*, Lecture Notes in Computer Science 1500, Springer, Berlin, 1999.
4. Kellner, M.I., R.J. Madachy and D.M. Raffo. Software Process Simulation Modeling: Why? What? How?, *Journal of Systems and Software*, 46, 2-3, 91-105, 1999.
5. Paulk, M.C., B. Curtis, M.B. Chrissis and C.V. Weber. The Capability Maturity Model for Software, Version 1.1, *IEEE Software*, 10, 4, 18-27, 1993.
6. Pfahl, D. and G. Ruhe. IMMoS: A Methodology for Integrated Measurement, Modeling, and Simulation, *Software Process Improvement and Practice*, 7, 189-210, 2002.