SOFTWARE PROCESS DYNAMICS

IEEE Press
445 Hoes Lane
Piscataway, NJ 08854

Technical Reviewers
Raman Aravamudham, University of Iowa
Márcio Barros, UNIRIO/Brazil
Guilherme H. Travassos, COPPE/Federal University of Rio de Janeiro, Brazil

SOFTWARE PROCESS DYNAMICS

Raymond J. Madachy

IEEE PRESS

WILEY-INTERSCIENCE
A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data is available.

ISBN 978-0-471-27455-1

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

CONTENTS

FOREWORD

The pace of change in software-intensive systems continues to accelerate at a dizzying rate. This presents a huge challenge for people trying to develop useful software. In the early days of software development, developers could freeze the requirements for the software, develop the software to the requirements, and deliver the resulting software two years later with confidence that the requirements would still be relevant and the software would be useful. Most of our software engineering processes, methods, and tools were developed and used under the assumption of relatively stable requirements. Examples are formal specification languages, performance-optimized point-solution designs, fixed-requirements software-cost estimation, earned-value management systems, requirements traceability matrices, fixed-price/fixed-requirements contracts, and a general attitude that "requirements creep" was bad in that it destabilized software development.

However, as these practices became increasingly institutionalized, the accelerating rate of software change made them increasingly risky to use. Projects would use them for two years and become extremely frustrated when the users were not interested in the obsolete capabilities that resulted. Projects would fall behind schedule and use static models (time to complete = work remaining divided by work rate) to try to make up time by adding people, and run afoul of Brooks's law (adding people to a late software project will make it later). Or they would sprint for the finish line using a point-solution design that satisfied the initial requirements but was extremely difficult to modify when trying to satisfy users' changing requirements.

Ironically, even with all of these difficulties, organizations would increasingly turn to software and its ability to be electronically upgraded as their best way to adapt their products, services, and systems to the increasing pace of change in their business or operational environment.

In order to keep up with this increased demand for software and the rapid pace of change, software organizations and projects need better ways to reason about the effects of change on their software products, projects, and processes. This is often very difficult to do, as software changes often have complex second-order and higher-order interaction effects that are hard to visualize and reason about. Thus, for example, a project with high rates of change in its requirements, being developed by a team with high rates of change in its personnel, will need to understand and control the interactions of decreased productivity due to change processing, decreased productivity due to new staff members' unfamiliarity with the project and product, and loss of productivity when key staff members are diverted from project work to train and mentor new people in order to develop increased downstream productivity.

One of the best techniques for reasoning about the effects of such complex interacting changes is the System Dynamics modeling framework that Ray Madachy presents in this book. As I have found in numerous applications of the method, it enables project personnel to model such effects and run the models to better understand the implications of candidate project strategies and decisions.

From the pioneering application of Jay Forrester's System Dynamics approach to software project modeling by Tarek Abdel-Hamid and Stuart Madnick in their 1991 book *Software Project Dynamics,* system dynamics modeling has been applied to many aspects of software development and management. These include the analysis of the effects on software system cost, schedule, and quality of alternative approaches to software process sequencing, software requirements determination, software architecture and design, software project organization and staffing, software development and integration, software quality assurance, and software change management. These applications have constituted the major source of solution approaches in the software process simulation community and its annual series of ProSim conferences (recently combined with the Software Process Workshop into the International Conference on Software Process, held concurrently with the International Conference on Software Engineering).

Ray Madachy has been a major contributor to this body of work. His experience in applying system dynamics modeling as a technical leader in such diverse organizations as the aerospace corporation Litton Systems, the e-commerce system development company C-Bridge Institute, and the software tools company Cost Xpert Group have given him a broad and deep perspective on the critical success factors of developing and applying system dynamics models to various classes of software decision situations. His experience in teaching and researching these techniques at USC has enabled him to develop an integrating framework and set of techniques that make system dynamics modeling much easier and cost-effective to learn and apply to a software decision situation.

His resulting book begins with an overview of the systems dynamics modeling framework, and an example application showing how to develop a system dynamics model that helps explain the conditions under which you can add people to a software project with or without having Brooks's Law apply. Next is an extensive chapter on the modeling process that introduces concepts and techniques used to develop system dynamics models, with illustrations of how they are developed and applied. The next

chapter provides and explains a full range of model structures from modeling molecules to large infrastructures and flow chains that can be used in new models. Once this framework is established, there are three chapters that apply it to the most significant classes of system dynamics applications.

The first of these chapters covers useful applications to software personnel decisions: workforce levels and team composition, learning, burnout, skills, motivation, retention, and rotation effects. The second chapter covers software process and product decision situations: peer reviews, software reuse, COTS-based system development, software architecting, and software process improvement effects. The third chapter covers project and organization applications, such as business case analysis, defect reduction strategies, and staffing management strategies. The final chapter projects likely future uses and research challenges in software process modeling and simulation, including how system dynamics models can add value via integration with other classes of models.

The appendices also provide important and one-of-a-kind material. The first appendix covers statistics of simulation, which is not covered in traditional references on system dynamics. Next is a thorough annotated bibliography of work using system dynamics for software processes, and is the definitive compendium for the field. Finally, the last appendix lists executable models provided with the book. These are used in examples and can be used for further exercises or for incorporation into your own models. These are valuable, reusable, and fun assets to go along with the book.

Overall, the book brings together a tremendous amount of useful process modeling material and experience in using it in practical software decision situations. It organizes this material into a unifying framework that makes it easier to apply and explain, and illustrates it with a wide variety of useful examples. I believe that the book will serve as a standard reference for the software process dynamics field and a great help to practitioners and researchers for a good long time.

BARRY BOEHM
University of Southern California

Los Angeles, California
June 2006

PREFACE

This book is designed for professionals and students in software engineering or information technology who are interested in understanding the dynamics of software development, or in assessing and optimizing process strategies. Its purpose is to improve decision making about projects and organizational policies by making its readers better informed about the dynamic consequences of decisions. Decisions may involve setting project budgets and schedules; return-on-investment analysis; trade-offs between cost, schedule, and quality or other factors; personnel hiring; risk management decisions; make, buy, or reuse; process improvement strategies; and so on.

The importance of process dynamics is hard to refute given the well-known (but too often ignored) combined effects of schedule pressure, communication overhead, changing business conditions, requirements volatility and user requests, experience, work methods such as reviews and quality assurance activities, task underestimation, bureaucratic delays, organizational shifts, demotivating events, other sociotechnical phenomena, and the feedback therein. These complex and interacting process effects are elegantly modeled with system dynamics using continuous quantities interconnected in loops of information feedback and circular causality. Knowledge of the interrelated technical and social factors coupled with simulation tools can provide a means for organizations to improve their processes.

The objectives of this book are to:

- Provide methods, tools, models, and examples to improve management decision making at all levels. Simulation can support corporate strategy and investment analysis, business case development, project and portfolio management, and training, for example.

- Illustrate *systems thinking* in action to develop increasingly deep understandings of software process structures and behaviors.
- Describe the modeling process, including calibration of models to software metrics data.
- Show basic building blocks and model infrastructures for software development processes.
- Review the field of software process modeling with system dynamics. Show how others have used the principles of system dynamics to analyze and improve processes in organizations.
- Provide practical lessons learned about the dynamics of software processes.
- Provide sufficient introductory material, including exercises and executable models on the Internet. Software practitioners who are brand new to simulation can immediately get their hands dirty and start modeling. Students can learn at their own pace, delving into the models as deeply as time and interest dictate.
- For those experienced in software process simulation, provide more detail of critical implementation issues and future research motivations.

The book is mostly new material, except for some example applications, and synthesizes previous work in the area. There has been much growth in the field; it has evolved to a state of maturity, and this book addresses the need to communicate findings. It draws from over 100 publications from practitioners and researchers experienced in system dynamics modeling of processes in organizations (all of them summarized in Appendix B). It is written to be a self-contained learning experience, and a comprehensive reference for modeling practitioners. The sections are structured so that readers can approach the subject from different perspectives and gain valuable knowledge for further study and practice depending on their needs.

A constructive understanding of process dynamics is provided by the illustrated models. Where appropriate, guidelines are presented for process improvement and general software management strategies (common threads include risk management and concentrating on people). The perspective in the book addresses the *dynamics* of software development, and best practices are described from that view. Some of these practices are illuminated through simulation experiments herein, and some will become foci of further study.

Readers may be involved in software process improvement, project planning and management, software development, testing, quality assurance, strategic corporate planning, organizational learning, education, or simply desire to understand the interrelated factors of software development. There is no need for sophisticated math skills, but a passing knowledge of numerical integration concepts will make the introductory material easier. Readers will increase their understanding of the complexities of software development and be able to use system dynamics for modeling and improving processes in their particular organizations. They will gain insight into the real-world mechanics behind the modeling equations.

For academic uses, this book may serve as an upper-level or graduate textbook for Software Process Modeling or other simulation courses. It can be used to support cur-

riculums in Software Engineering, Software Project Management, Software Quality Assurance, Systems Engineering or related subjects.

Part 1 of the book presents modeling fundamentals for software processes. These chapters may constitute an entire course for those new to system dynamics modeling. Advanced students should cover applications and future directions in Part 2. These topics can be studied in whole or as selected subjects. Other disciplines or focused studies may choose relevant application topics. For example, researchers in organizations or sociology might want to cover people applications, engineering process groups might investigate selected process applications, while senior managers could focus on project or organizational applications.

The sequence and depth of subjects should be tailored accordingly for any of these uses. The variety of exercises in the book may serve as homework assignments, exam questions, or even major research projects. Except for the introductory chapters, detailed equations are generally omitted from the text and left for the reader to study from the models.

Though a primary objective is to instruct in computer-aided analysis, several identified exercises early in the book should be done without a computer. This is to help develop intuition of process dynamics, and to strike a balance by not becoming overly reliant on blindly using the computer during model development and evaluation of simulation results. The structure of the book is explained in more detail below.

BOOK ORGANIZATION AND HIGHLIGHTS

This section provides a sequential outline of topics with selected highlights. Each chapter includes numerous graphics, charts, and tables to help illustrate the material. The book is also supplemented on the Internet containing the sample models and simulation tools, exercises, extra references, and updates to the material. The book is divided into two major parts per the outline below:

Part 1 Fundamentals
- Chapter 1—Introduction and Background
- Chapter 2—The Modeling Process with System Dynamics
- Chapter 3—Model Structures and Behaviors for Software Processes

Part 2 Applications and Future Directions
- Introduction to Applications Chapters
- Chapter 4—People Applications
- Chapter 5—Process and Product Applications
- Chapter 6—Project and Organization Applications
- Chapter 7—Current and Future Directions

Appendices and References
- Appendix A—Introduction to Statistics of Simulation
- Appendix B—Annotated Bibliography
- Appendix C—Provided Models
- References

Chapter 1 establishes the context and foundation of the book, with a goal of helping people use models to quantitatively evaluate processes in order to make better decisions. The chapter presents an introduction and background including motivational issues and a capsule history of the field. Definitions of terms are provided for reference throughout the book. The concepts of systems thinking are introduced, so one can see how simulation can be used to leverage learning efforts and improve organizational performance. Control systems principles are introduced, and then a simple motivational example of modeling Brooks's Law is shown. A review of software process technology covers process modeling, lifecycle models, and process improvement.

A description of the iterative modeling process with the system dynamics simulation methodology is provided in Chapter 2. Basic modeling elements and classical system behaviors are shown. The underlying mathematical formulation of system dynamics is covered with its ramifications for software process models.

The activities of problem definition, model formulation (including calibration), simulation, assessment, communication to others, and challenging the model for the next iteration are elaborated on. Since simulation is both art and science, guidelines and modeling heuristics are discussed. It is seen that there is much in common with software engineering principles in general such as iteration, abstraction, aggregation, and so on, yet there are also aspects of simulation that require somewhat different skills.

This chapter also details the multiperspective validation of system dynamics models, which is of paramount importance before drawing policy conclusions from simulation experiments. Different modeling tools and environments are overviewed to help modelers in choosing appropriate tools for their different needs. Also see Appendix A on the use of statistics in the modeling process.

Chapter 3 presents patterns of model structures and behaviors for software processes. Included is a detailed description of levels, flows, auxiliaries, infrastructures, and feedback loops instantiated for software processes. State variables of interest include software work artifacts, defect levels, personnel levels, effort expenditure, schedule date, and others. Corresponding rates over time include software productivity, defect introduction and elimination rates, financial flows for costs and revenue, and so on. Project reference behaviors for different structures and management policies are introduced.

An important contribution of this chapter is the explication of basic flow processes for software development. Common structures for software processes ferreted out of the major models (upcoming in Chapters 4 through 6) are shown. Together with prototypical feedback loops such as learning and project controlling, these infrastructures can be (re)used to develop models relevant to any software process. This section also illustrates a major advantage in system dynamics models over other modeling techniques: inherent cost, schedule, and quality trade-offs by modeling their interactions.

Part 2 covers modeling applications in the field and future directions. Chapter 4 focuses on people applications, Chapter 5 covers process and product applications, and Chapter 6 is about projects and organizations. Each chapter contains applications of varying complexity. An overview of applications and research to date is provided, including history, a list of different implementations, and critiques of the various work.

Modeling examples from the field are shown with sample insights. The examples are further instances of the generic structures from Chapter 3.

The application examples show threads of simulation modeling with actual model implementations and worked out examples. These original examples should be of particular value to system dynamics novices, and more experienced modelers can study them for additional ideas. Many also amplify some lessons learned regarding the software process. Some of the example models are also contained on the accompanying website. Additional exercises are provided for students to work out and practitioners to implement. Note that the applications chapters will also be updated online to keep up with new work.

Chapter 7 presents current and future directions in software process modeling and simulation. These include advances in simulation environments and tools, model structures and component-based model development, new and emerging trends for application models, model integration (not just system dynamics models), empirical research, theory building, and putting it all together in process mission control centers and training facilities.

Appendix A introduces statistics for simulation as an addendum to the modeling fundamentals about which simulation analysts, researchers, and graduate students studying broader aspects of simulation must be knowledgeable. Statistical methods are used to handle the stochastic inputs and outputs of simulation models. The appendix covers the principles of probability distributions, sample size, confidence intervals, and experimental design applied to continuous system simulation. Monte Carlo simulation is described and recommended probability distributions for software process modeling are also provided.

Appendix B is an annotated bibliography of using system dynamics for software processes and is the most complete set of references for the field. It demonstrates well the breadth of applications to date and is a convenient place to start researching particular topics. These same citations are identified in the References in boldface.

Appendix C lists the provided models referenced in the chapters or exercises. These go along with the examples, and can be executed and modified by readers for their own purposes. These models will be updated and replaced on the Internet as improvements are made. Models provided by other readers will also be posted.

INTERNET SITES

The referenced models, tools, updates, discussion, and color book information are available on the world wide web at http://csse.usc.edu/softwareprocessdynamics and at a mirror site http://softwareprocessdynamics.org.

ACKNOWLEDGMENTS

I would like to extend sincere appreciation to all the other people who contributed to this work. I initially learned system dynamics for physiological modeling in a graduate

biomedical engineering course at UCSD in 1982 under the excellent direction of Drs. Alan Schneider and James Bush. This book would not be complete without the accomplishments of other researchers and support of colleagues including Dr. Tarek Abel-Hamid, Richard Adams, Dr. Vic Basili, Dr. James Collofello, Scott Duncan, Dr. Susan Ferreira, Dr. David Ford, Tobias Haberlein, Jim Hart, Dr. Dan Houston, Dr. Marc Kellner, Peter Lakey, Dr. Manny Lehman, Dr. Robert Martin, Dr. Margaret Johnson, Emily oh Navarro, Dr. Nathaniel Osgood, Dr. Dietmar Pfahl, Oliver Pospisil, Dr. David Raffo, Dr. Juan Ramil, Dr. Stan Rifkin, Dr. Howard Rubin, Dr. Ioana Rus, Dr. Walt Scacchi, Dr. Neil Smith, Dr. Greg Twaites, Dr. Wayne Wakeland, Dr. Gerry Weinberg, Dr. Paul Wernick, and Ed Yourdon; Litton personnel including Dr. Denton Tarbet, Wayne Sebera, Larry Bean, Frank Harvey, and Roy Nakahara; Charles Leinbach from C-bridge Institute; Benny Barbe from Cost Xpert Group; and Dr. Julian Richardson and Dr. Michael Lowry for their support at NASA. USC graduate students who contributed to this work are Ashwin Bhatnagar, Cyrus Fakharzadeh, Jo Ann Lane, Dr. Nikunj Mehta, Kam Wing Lo, Jason Ho, Leila Kaghazian, Dr. Jongmoon Baik (also including post-graduate contributions), and Wook Kim. Profound thanks goes to Dr. Barry Boehm, who has served as a mentor and been my biggest influence since the middle of my Ph.D. studies. This book owes much to his continual support, penetrating insights, and inspiration to contribute. Many thanks to the anonymous IEEE reviewers for their long hours and detailed constructive reviews, and IEEE staff including Jeanne Audino, Cathy Faduska, Chrissy Kuhnen, Cheryl Baltes, and Matt Loeb. I also am most grateful to my wife Nancy for her long-term support and lots of patience, and my young daughters Zoey and Deziree for extra motivation and lessons on adaptation to change.

BOOK UPDATES AND MAKING CONTRIBUTIONS

The field of software process modeling itself is quite dynamic, with much happening in conjunction with other software process work. It has been a challenge keeping up with the times as this book has progressed, and the rate of change in the industry has increased over these years. It is inevitable that some things will continue to change, so the reader is urged to access the Internet site for updates at any time, including new and improved models.

Updates to the chapters will be put on the book's Internet site until the next published edition. The application Chapters 4–6 will have substantial updates and entire sections replaced. The goal is to keep the applications current and presented in a uniform format. Chapter 7 on current and future directions is a wild card in terms of predicted changes, and the annotated bibliography will be updated continuously.

It is an exciting time with much opportunity and more work to be done. Hopefully, some of the ideas and exercises in this book will be used as a basis for further practice and research. People will provide new and better exercises and those will be posted too. Your comments on this book and experiences with modeling actual processes are of great interest to this author, and your feedback will help in developing the next edi-

tion. You are encouraged to send any ideas, improvement suggestions, new and enhanced models, or worked out exercises from this book. They will be included in future editions as appropriate.

RAYMOND J. MADACHY

Los Angeles, California
November 2007