国外优秀信息科学与技术系列教学用书

软件项目管理

——统一性框架

（影印版）

SOFTWARE PROJECT MANAGEMENT

A Unified Framework

■ Walker Royce

高等教育出版社
Higher Education Press
Pearson Education
出版集团

国外优秀信息科学与技术系列教学用书

软件项目管理

——统一性框架

（影印版）

SOFTWARE PROJECT MANAGEMENT

A Unified Framework

Walker Royce

高等教育出版社

Pearson Education 出版集团

图字：01-2002-3774 号

Software Project Management: A Unified Framework, First Edition
Walker Royce

图书在版编目(CIP)数据

软件项目管理：统一性框架/（美）罗伊斯（Royce，W.）著. —影印本. —北京：高等教育出版社，2002. 10
ISBN 7-04-011397-X

Ⅰ. 软… Ⅱ. 罗… Ⅲ. 软件开发-项目管理-高等学校-教材-英文 Ⅳ. TP311.52

中国版本图书馆 CIP 数据核字(2002)第 081695 号

软件项目管理——统一性框架 (影印版)

Walker Royce

出版发行	高等教育出版社	**购书热线**	010-64054588
社　　址	北京市东城区沙滩后街 55 号	**免费咨询**	800-810-0598
邮政编码	100009	**网　　址**	http://www.hep.edu.cn
传　　真	010-64014048		http://www.hep.com.cn
经　　销	新华书店北京发行所		
印　　刷	北京中科印刷有限公司		
开　　本	787×1092　1/16	**版　　次**	2002 年 10 月第 1 版
印　　张	27.25	**印　　次**	2002 年 10 月第 1 次印刷
字　　数	670 000	**定　　价**	30.00 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

出 版 说 明

20 世纪末，以计算机和通信技术为代表的信息科学和技术对世界经济、科技、军事、教育和文化等产生了深刻影响。信息科学技术的迅速普及和应用，带动了世界范围信息产业的蓬勃发展，为许多国家带来了丰厚的回报。

进入 21 世纪，尤其随着我国加入 WTO，信息产业的国际竞争将更加激烈。我国信息产业虽然在 20 世纪末取得了迅猛发展，但与发达国家相比，甚至与印度、爱尔兰等国家相比，还有很大差距。国家信息化的发展速度和信息产业的国际竞争能力，最终都将取决于信息科学技术人才的质量和数量。引进国外信息科学和技术优秀教材，在有条件的学校推动开展英语授课或双语教学，是教育部为加快培养大批高质量的信息技术人才采取的一项重要举措。

为此，教育部要求由高等教育出版社首先开展信息科学和技术教材的引进试点工作。同时提出了两点要求，一是要高水平，二是要低价格。在高等教育出版社和信息科学技术引进教材专家组的努力下，经过比较短的时间，第一批由教育部高等教育司推荐的 20 多种引进教材已经陆续出版。这套教材出版后受到了广泛的好评，其中有不少是世界信息科学技术领域著名专家、教授的经典之作和反映信息科学技术最新进展的优秀作品，代表了目前世界信息科学技术教育的一流水平，而且价格也是最优惠的，与国内同类自编教材相当。这套教材基本覆盖了计算机科学与技术专业的课程体系，体现了权威性、系统性、先进性和经济性等特点。

目前，教育部正在全国 35 所高校推动示范性软件学院的建设，这也是加快培养信息科学技术人才的重要举措之一。为配合软件学院的教学工作，结合各软件学院的教学计划和课程设置，高等教育出版社近期聘请有关专家和软件学院的教师遴选推荐了一批相应的原版教学用书，正陆续组织出版，以方便各软件学院开展双语教学。

我们希望这些教学用书的引进出版，对于提高我国高等学校信息科学技术的教学水平，缩小与国际先进水平的差距，加快培养一大批具有国际竞争力的高质量信息技术人才，起到积极的推动作用。同时我们也欢迎广大教师和专家们对我们的教材引进工作提出宝贵的意见和建议。联系方式：hep.cs@263.net。

高等教育出版社

二〇〇二年九月

This work is dedicated to my father, Winston Royce, whose vision and practicality were always in balance.

—Walker

Foreword

This book blazes the way toward the next generation of software management practice. Many organizations still cling to the waterfall model because, even with its shortfalls, it provides the most fully elaborated management guidelines on how to proceed in a given software situation.

It has been difficult to find a fully articulated alternative management approach for dealing with such issues as commercial component integration, software reuse, risk management, and evolutionary/incremental/spiral software processes. This book provides a new experience-tested framework and set of guidelines on how to proceed.

Walker Royce developed and tested this software management approach during his inception-to-delivery participation in the large, successful CCPDS-R project performed by TRW for the U.S. Air Force. He then refined and generalized it across a wide spectrum of government, aerospace, and commercial software development experiences at Rational.

Chapters 1 through 4 of the book motivate the approach by showing how it gives you management control of the key software economics leverage points with respect to traditional software management. These are (1) reducing the amount of software you need to build, (2) reducing rework via improved processes and teamwork, and (3) reducing the labor-intensiveness of the remaining work via automation.

Chapters 5 through 10 present the specifics of a new organization of the software life cycle, which also forms the management basis for Rational's Unified process. It combines the flexibility of the spiral model with the discipline of risk management and a set of major life-cycle phases and milestones. These milestones are focused on major management commitments to life-cycle courses of action.

As with our Anchor Point approach at USC, the life-cycle objectives milestone involves a management commitment to engage in a software architecting effort based on a business case analysis (or not to engage, in which case the project is mercifully

killed). The life-cycle architecture milestone involves a management commitment to proceed into full-scale development based on establishing and demonstrating a sound architecture and resolving all major risk items. The initial operational capability milestone involves a management commitment to proceed to beta testing the product with outside users, or its equivalent.

In these chapters, Royce provides a set of views showing how these milestones differ from conventional document-oriented or code-oriented milestones. Instead, the key product artifact sets (requirements, design, implementation, deployment) concurrently evolve and coalesce in a manner consistent with the project's objectives and its strategies for controlling risk.

In Chapters 10 through 14, Royce addresses how to ensure that the software project's management artifacts are also concurrently evolving and coalescing. These include the project's plans and associated cost and schedule estimates, the project's organization and team-building activities, and the project's metrics, instrumentation, and control processes. Chapter 14 is particularly noteworthy. It not only emphasizes that the management solutions are situation-dependent, it also provides guidelines for tailoring them to the project's scale, team culture, process maturity, architectural risk, and domain experience.

In Chapters 15 through 17, Royce looks forward to where the best software developers are going with their practices: toward product line management, round-trip engineering, and smaller teams with managers as performers and quality assurance as everyone's job. Appendixes relate his software management approach to the current state of the practice, to the COCOMO and COCOMO II family of cost models, and to the SEI Capability Maturity Model. Appendix D provides a convincing case study of how the approach was successfully used on the large, technically challenging CCPDS-R project.

Royce has a refreshing candor about some of the fads, follies, and excesses in the software field. This comes out particularly in several "pragmatic" sections that address such topics as software cost estimation, inspections, artifacts, planning, and metrics. Not everyone will agree with all of his assessments, particularly on inspections, but they are incisive and thought-provoking.

I feel extremely fortunate to have been able to work with both Walker Royce and his equally insightful father, Winston Royce; to have learned from their experiences; and to have interacted with them as they evolved their path-breaking ideas.

Barry Boehm
Director, USC Center for Software Engineering
April 1998

Preface

The software industry moves unrelentingly toward new methods for managing the ever-increasing complexity of software projects. In the past, we have seen evolutions, revolutions, and recurring themes of success and failure. While software technologies, processes, and methods have advanced rapidly, software engineering remains a people-intensive process. Consequently, techniques for managing people, technology, resources, and risks have profound leverage.

This book captures a software management perspective that emphasizes a balanced view of these elements:

- Theory and practice
- Technology and people
- Customer value and provider profitability
- Strategies and tactics

Throughout, you should observe a recurring management theme of paramount importance: *balance.* It is especially important to achieve balance among the objectives of the various stakeholders, who communicate with one another in a variety of languages and notations. Herein is the motivation for the part opener art, an abstract portrayal of the Rosetta stone. The three fundamental representation languages inherent in software engineering are requirements (the language of the problem space), design (the transformation languages of software engineers), and realizations (the language of the solution space executable on computers). Just as the Rosetta stone enabled the translation of Egyptian hieroglyphics, software management techniques enable the translation of a problem statement into a solution that satisfies all stakeholders.

There is no cookbook for software management. There are no recipes for obvious good practices. I have tried to approach the issues with as much science, realism, and experience as possible, but management is largely a matter of judgment, (un)common sense, and situation-dependent decision making. That's why managers are paid big bucks.

Some chapters include sections with a pragmatic and often hard-hitting treatment of a particular topic. To differentiate this real-world guidance from the general process models, techniques, and disciplines, headings of these sections include the word *pragmatic*. By pragmatic I mean having no illusions and facing reality squarely, which is exactly the intent of these sections. They contain strong opinions and provocative positions, and will strike nerves in readers who are entrenched in some obsolete or overhyped practices, tools, or techniques.

I have attempted to differentiate among proven techniques, new approaches, and obsolete techniques using appropriate substantiation. In most cases, I support my positions with simple economic arguments and common sense, along with anecdotal experience from field applications. Much of the material synthesizes lessons learned (state-of-the-practice) managing successful software projects over the past 10 years. On the other hand, some of the material represents substantially new (state-of-the-art), hypothesized approaches that do not have clear substantiation in practice.

I have struggled with whether to position this book as management *education* or management *training*. The distinction may seem nitpicky, but it is important. An example I heard 15 years ago illustrates the difference. Suppose your 14-year-old daughter came home from school one day and asked, "Mom and Dad, may I take the sex education course offered at school?" Your reaction would likely be different if she asked, "May I take the sex training course offered at school?" (This meant less to me then than it does now that my three daughters are teenagers!)

Training has an aspect of applied knowledge that makes the knowledge more or less immediately useful. Education, on the other hand, is focused more on teaching the principles, experience base, and spirit of the subject, with the application of such knowledge left to the student. I have tried to focus this book as a vehicle for software management education. (I am not sure there is such a thing as management training other than on-the-job experience.) I will not pretend that my advice is directly applicable on every project. Although I have tried to substantiate as many of the position statements as possible, some of them are left unsubstantiated as pure hypotheses. I hope my conjecture and advice will stimulate further debate and progress.

My intended audience runs the gamut of practicing software professionals. Primary target readers are decision makers: those people who authorize investment and expenditure of software-related budgets. This group includes organization managers, project managers, software acquisition officials, and their staffs. For this audience, I am trying to provide directly applicable guidance for use in today's tactical decision

making and tomorrow's strategic investments. Another important audience is software practitioners who negotiate and execute software project plans and deliver on organizational and project objectives.

Style

Because I am writing for a wide audience, I do not delve into technical perspectives or technical artifacts, many of which are better discussed in other books. Instead, I provide fairly deep discussions of the economics, management artifacts, work breakdown strategies, organization strategies, and metrics necessary to plan and execute a successful software project.

Illustrations are included to make these complex topics more understandable. The precision and accuracy of the figures and tables merit some comment. While most of the numerical data accurately describe some concept, trend, expectation, or relationship, the presentation formats are purposely imprecise. In the context of software management, the difference between precision and accuracy is not as trivial as it may seem, for two reasons:

1. Software management is full of gray areas, situation dependencies, and ambiguous trade-offs. It is difficult, if not impossible, to provide an accurate depiction of many concepts *and* to retain precision of the presentation across a broad range of domains.
2. Understanding the difference between precision and accuracy is a fundamental skill of good software managers, who must accurately forecast estimates, risks, and the effects of change. Unjustified precision—in requirements or plans—has proven to be a substantial, yet subtle, recurring obstacle to success.

In many of my numeric presentations, the absolute values are unimportant and quite variable across different domains and project circumstances. The relative values constitute the gist of most of the figures and tables.

I occasionally provide anecdotal evidence and actual field experience to put the management approaches into a tangible context and provide relatively accurate and precise benchmarks of performance under game conditions. Several appendixes clarify how the techniques presented herein can be applied in real-world contexts. My flagship case study is a thoroughly documented, successful, large-scale project that provides a concrete example of how well many of these management approaches can work. It also provides a framework for rationalizing some of the improved processes and techniques.

Organization

The book is laid out in five parts, each with multiple chapters:

- **Part I, Software Management Renaissance.** Describes the current state of software management practice and software economics, and introduces the state transitions necessary for improved software return on investment.
- **Part II, A Software Management Process Framework.** Describes the process primitives and a framework for modern software management, including the life-cycle phases, artifacts, workflows, and checkpoints.
- **Part III, Software Management Disciplines.** Summarizes some of the critical techniques associated with planning, controlling, and automating a modern software process.
- **Part IV, Looking Forward.** Hypothesizes the project performance expectations for modern projects and next-generation software economics, and discusses the culture shifts necessary for success.
- **Part V, Case Studies and Backup Material.** Five appendixes provide substantial foundations for some of the recommendations, guidance, and opinions presented elsewhere.

Acknowledgments

Although my perspective of iterative development has been influenced by many sources, I have drawn on relatively few published works in writing this book. Providing a more detailed survey of related publications might have helped some readers and satisfied some authors, but most of the correlation with my views would be coincidental.

The foundation of my material comes basically from three sources, on which I have drawn extensively:

1. TRW's *Ada Process Model Guidebook* [Royce, Walker, 1989]. I wrote this guidebook to capture the process description implemented successfully on a large-scale TRW project so that it could be used throughout TRW.
2. Rational Software Corporation's software management seminar [Royce, Walker, 1997]. I wrote this two-day seminar on software best practices to describe Rational's software management approach. The peer reviewers for this material included Don Andres (TRW), Barry Boehm (University of Southern California), Larry Druffel (Software Engineering Institute), Lloyd Mosemann (U.S. Air Force), and Winston Royce (TRW), in addition to numerous field practitioners and executives within Rational. The seminar was delivered dozens of times in the mid-1990s to a broad range of audiences, including government groups, defense contractors, and commercial organizations.

3. Rational's Unified process. The acquisition of Objectory by Rational resulted in a large internal investment to merge the techniques of the Objectory process (focused on use-case-driven techniques) and the existing Rational process (focused on management techniques and object-oriented modeling). This investment is on-going, as Rational continues to broaden the process description and prescription across more of the life-cycle activities, tools, and methods, resulting in the Unified process.

Several other sources had a significant effect on the management process presented in this book. Their influence is the result of long-term relationships that encapsulate years of interaction, exchange of ideas, and extensive firsthand communication.

- My association with Barry Boehm over the past 15 years has been a rich source of software engineering knowledge.
- Don Andres's extraordinary leadership and project management expertise set him apart from the many project managers I have worked for and with, and I have learned much from him.
- Dave Bernstein, Robert Bond, Mike Devlin, Kevin Haar, Paul Levy, John Lovitt, and Joe Marasco, senior managers at Rational, have evolved a nimble company with a clear vision of software engineering as a business.
- Philippe Kruchten's work on software architecture and process frameworks, as well as his own field experience, has helped gel many of my perspectives and presentations.
- Grady Booch, Ivar Jacobson, and Jim Rumbaugh, Rational's three senior methodologists, have done the software engineering community a great service in defining the Unified Modeling Language.
- Hundreds of dedicated software professionals in the Rational field organization have been responsible for delivering value to software projects and transitioning software engineering theory into practice.

The most important influence on this work was my father, Winston Royce, who set my context, validated my positions, critiqued my presentation, and strengthened my resolve to take a provocative stand and stimulate progress.

Several people invested their own time reviewing early versions of my manuscript and contributing to the concepts, presentation, and quality contained herein. My special thanks go to Ali Ali, Don Andres, Peter Biche, Barry Boehm, Grady Booch, Doug Ishigaki, Ivar Jacobson, Capers Jones, Hartmut Kocher, Philippe Kruchten, Eric Larsen, Joe Marasco, Lloyd Mosemann, Roger Oberg, Rich Reitman, Jim Rumbaugh, and John Smith.

Finally, the overall presentation quality, consistency, and understandability of this material are substantially the work of Karen Ailor. Her critique, sense of organization, attention to detail, and aggressive nitpicking contributed greatly to the overall substance captured in this book.

责任编辑　康兆华
封面设计　张　楠
责任印制　宋克学

郑 重 声 明

Contents