软件工程与方法丛书

影印版

软件项目管理

Software Project Management

A Real-World Guide to Success

- Get help organizing the suggested approaches
- Demystify software engineering concepts that may seem abstract
- Figure out which tools are available to put management practices into use

(美) Joel Henry 编著

Pearson Education

科学出版社
www.sciencep.com

软件工程与方法丛书

软件项目管理

（影印版）

Software Project Management:
A Real-World Guide to Success

（美） Joel Henry 编著

科学出版社

北京

图字：01-2003-7662 号

内 容 简 介

本书全面介绍了软件项目管理的概念、理论和实务，按照项目进行的流程，以简明的语言、典型的案例给出软件项目管理的实施方法，将抽象的软件工程概念具体化，并且提供了可以在管理实务中应用的各种工具。本书具有较高的学术参考价值，适合作为高等学校研究生或本科生“软件项目管理”课程的教材，也可供软件企业在开发和管理人员培训时使用。

图书在版编目(CIP)数据

软件项目管理= Software Project Management: A Real-World Guide to Success /（美）亨利（Joel Henry）编著.—影印版. —北京：科学出版社，2004
（软件工程与方法丛书）
ISBN 7-03-012496-0
Ⅰ.软… Ⅱ.亨… Ⅲ.软件开发—项目管理—英文 Ⅳ.TP311.52

中国版本图书馆 CIP 数据核字（2003）第 103016 号

策划编辑：李佩乾/责任编辑：袁永康
责任印制：吕春珉/封面设计：飞天创意

科学出版社 出版
北京东黄城根北街16号
邮政编码:100717
http://www.sciencep.com

双青印刷厂 印刷

科学出版社发行 各地新华书店经销
*
2004 年 1 月第 一 版 开本：787×960 1/16
2004 年 1 月第 一次印刷 印张：27 1/4
印数：1—3 000 字数：420 000

定价：45.00 元

（如有印装质量问题，我社负责调换〈环伟〉）

影印前言

“软件工程”是自20世纪60年代起针对所谓“软件危机”而发展起来的概念。它是指将工程化的方法应用到软件开发中，以求优质高效地生产软件产品。其中综合应用了计算机科学、数学、工程学和管理科学的原理和方法。自从这一概念提出以来，软件开发方法从60年代毫无工程性可言的手工作坊式开发，过渡到70年代的结构化分析设计方法、80年代初的实体关系方法，直到当今所流行的面向对象方法，经历了根本性的变革。随着时代的发展，软件项目管理人员越来越需要从系统和战略的角度来把握项目的方向，引导开发向更高层次发展。在这方面，国外的知名企业和研究机构已经总结了相对成熟的一套知识和方法体系，并在实践中获得了相当大的成功。这里我们就从著名的培生教育出版集团（Pearson Education Group）选取了一些软件工程与方法类的有代表性的教材影印出版，以期让国内的读者尽快了解国外此领域的发展动态，分享国外专家的知识和经验。以下对每本书的内容作一些简要的介绍，以便读者选择。

在Internet广泛普及的今天，软件承载着越来越重要的使命，工程化的开发必须能够提供健壮性和普适性更好的产品。Scott E. Donaldson和Stanley G. Siegel合作编著的《成功的软件开发》（*Successful Software Development*）一书从“软件系统开发无定式”这一事实出发，引入了一个灵活而成熟的开发过程模型——系统工程环境（Systems Engineering Environment，SEE）。该模型包含两个互相联系的基本要素：确定软件开发策略和规程，以及可用于实现目标的技术。围绕这一模型，书中对开发过程中关系项目成败的各种关键问题进行了透彻的论述，可作为软件开发团队的全程培训教材。

软件工程经过几十年的发展，其关键环节——需求分析和管理终于得到了真正的重视。伴随认识的深化和案例的丰富，一系列实用的工程化需求分析方法应运而生。Ralph R. Young的《有效需求分析》（*Effective Requirements Practices*）一书从管理和技术两个角度阐述了关系项目成败的各种问题，书中的案例分析让项目管理者能够对需求分析的框架体系和过程形成较为清晰的认识，在实践中准确了解客户的业务需求，正确地调配各种资源，从而更加准确地把握项目的方向，保证在整个项目周期中各种需求都能够得到应有的考虑和满足。

软件开发人员在系统组织方面通常会采用特定的模式，但就大多数而言，他们对体系结构的分析和判断在很大程度上都是出于自发而且并不规范。Mary Shaw和David Garlan的《软件体系结构》（*Software Architecture: Perspectives on an Emerging Discipline*）一书就对当前业界所公认的成功的系统设计思想进行了生动而全面的阐述。两位作者对软件体系结构的发展状况及其对设计的影响作用有独到的见解，相信各类读者都能从书

中获得有用的信息：专业开发人员可能对书中所介绍的设计模式比较熟悉，但从对这些模式的讨论和评价中也能够获得新的启发；学生们能够从书中学到一些有用的技巧，从较高的视角来了解系统的组织结构，而不是仅仅追逐业界的潮流或是过时的方法；对于教师而言，本书可以作为“软件体系结构”课程的教材，或者是“软件工程”或“软件设计”课程的补充教材。

对于现在已经或者将要从事软件项目管理的读者来说，Joel Henry 的《软件项目管理》（*Software Project Management: A Real-World Guide to Success*）应该说是一本难得的好教材。书中论述了软件项目的四个基本构成要素：人、过程、工具和评价体系，并向读者提供每一领域中可以选用的合适方法，使项目实施取得最满意的效果。作者本人在软件行业工作多年，积累了丰富的第一手材料，他在书中用案例对技术、管理和领导等多方面问题进行了透彻的讲解。尽管他对这些问题的结论对业内人士而言已经是耳熟能详，但不论是何种类型和水平的读者，相信都能从本书中得到指导和启发。

软件开发中出现错误在所难免，关键是要及早发现，而不要让其扩散，增加纠正的成本。软件同级评审制度是任何高质量软件开发过程的重要环节，然而受过这方面必要培训的专业人员却还是凤毛麟角。Karl E. Wiegers 的《软件同级评审》（*Peer Reviews in Software*）就是针对这方面需求而编写的。本书介绍了软件同级评审的全过程，内容涉及各种正规和非正规的评审方法和质量保证技巧。书中对大型项目及开发团队地域分散等情况下的同级评审进行了专门探讨。对于项目管理者而言，本书能够帮助他们以实用的资源启动同级评审计划，增进开发人员间的沟通，最终按期提交高质量的软件产品。

面向对象技术的应用已经越来越普遍，而与此同时越来越多的管理者在项目进行中却要面对许多原本隐藏着的成本和意外情况。对整个项目而言，管理者在规划阶段是否有远见、在进行过程中对各种情况反应是否得当，这些都影响着项目的成败。尽管市场上介绍对象技术的书很多，但对于项目实施中所需进行的规划和预测却还缺乏系统的知识归纳。Alistair Cockburn 的《对象软件项目求生法则》（*Surviving Object-Oriented Projects*）一书就以大量专家的知识和经验向读者提供成功管理对象软件项目的实用指导和建议，帮助读者应对项目中的意外挑战，使项目正常进行并最终获得成功。书中指出了对象软件项目所面临的潜在风险，并对时间安排、预算、人员安排以及成本合理化等重要问题进行了探讨，提供了可操作的解决方案。对于从事对象软件项目管理的读者而言，本书是一本相当合适的参考书，可以作为相关培训的教材。

以上就是目前这套影印版丛书的大致内容。需要指出的是，这套丛书并非一个封闭的体系。随着软件工程的深入发展，必将涌现出更多新的开发理念和方法，我们也将尽己所能将更多新的优秀图书吸纳进来。读者对于这套丛书目前的内容或未来的发展有何意见或建议，望不吝赐之。

编　者

2003 年 11 月

Foreword

Often, knowing something about the author of a book helps put its content and the author's message into a clearer perspective. If the author has the knowledge and the experience it takes to be an expert in the field, it is easier to have confidence in what he or she has to say.

In the case of this book, this is particularly true. Joel has built on the shoulders of giants, but, as you will see when you read the book, many of the truths he has to convey are based on lessons he learned from trying ideas that did not work well.

I want to tell you a little about Joel Henry and his experience.

He earned his BS and MS degrees at Montana State University in 1985 and 1986, respectively. In 1993, he earned his PhD in Computer Science from Virginia Polytechnic Institute and State University. From the beginning, his research and special interests were in Software Engineering.

Fortunately for me, I was chair of the Department of Computer and Information Sciences at East Tennessee State University when Joel finished his doctoral studies, and our efforts to recruit him to join our faculty were successful. We always had a strong program, and when colleagues asked me to describe it, I often started by telling them about our "departmental superstar," Joel Henry.

Joel is a fantastic teacher; he combines his insistence on excellence with a sincere and deeply caring attitude in the way all teachers wish we could. No one ever passed one of his courses without learning the material in depth. Yet,

every student he ever had adored him as a teacher. Most of the great coaches in sports history have a special intensity that you can see in their eyes and that you can feel in their presence. They are loved by their players, but every one of the players knows instantly that total effort and commitment are a given if you want to play on their teams. Students feel the same way with Joel. His students have recognized his excellence through teaching awards, and his colleagues have recognized it through tenure, promotion, and imitation.

In addition to being a great teacher, Joel has produced more than two dozen technical publications and scholarly papers in various aspects of software engineering. Most of these focus on software testing and software process improvement. In spite of his scholarly accomplishments, he is not merely an academic researcher. His practical experience complements his academic background and achievements to make him a true expert in the field.

He has been involved in a number of major software projects for large government and industrial organizations including the Environmental Protection Agency, Lockheed-Martin, Digital Equipment Corporation, and Invensys. His software process evaluation and improvement experience is broad and includes significant projects with companies and agencies such as General Electric, the Defense Logistics Agency, the Software Engineering Institute, and NASA.

In the classroom, Joel builds on the lessons he has learned through his extensive industrial and governmental experiences. He teaches his students how to build quality software that is delivered on-time, within specifications, and within budget, and he emphasizes the importance of doing this in a team environment using well-documented, predictable, measurable, and repeatable process.

As one of Joel's colleagues, I always marveled at how he could teach his courses, advise his students, lead his teams, perform his research, and still find the time to consult and to write, in short, to *do* what he taught.

I have introduced you to the multifaceted person who has found the time to write this book. He has the education, the experience, and the professionalism to convince you that he knows what he is talking about and that he is leading you in the proper direction. He is extraordinary enough to have something to teach us while being ordinary enough so that we can identify with him and believe his methods can bring us success, too.

As you read, I hope you will think of the person as well as the professional who authored this book. You can trust him. He is a superb teacher, and all of us can learn from him.

Dr. Gordon Bailes
East Tennessee State University

Preface

Why read the preface of this book? This is where you find out what this book contains. You also begin to figure out who the author is, and how he came to write a book. Understanding this motivation is every bit as important as the content because it puts the book in context. A person doesn't wake up one day and decide "Gee, I think I'll sit down and type over 100,000 words, revise them a couple dozen times, wait 24 weeks for them to appear on the shelf, then hope someone buys the book so I'll know I wrote something someone actually read."

I wrote this book because I couldn't find a complete book to teach students and working professionals how to manage software projects. Much of the information was out there, spread throughout many books. These other books I found contained all kinds of excellent information but in none of them did I find everything I thought people should know about project management. There were the eclectic war stories of working software professionals and the formal, dictatorial, and documentation-heavy approaches of institute gurus. It seemed to me the majority of work in our field is not done by multibillion dollar companies or government contractors that can afford to learn and adhere to maturity models or prescribed software processes.

This book is the culmination of two decades of experience in academic and commercial software engineering. This doesn't make me one of the software engineering experts who run institutes or consulting firms; it makes me one of the guys who have sifted through research and practice,

and everything in between. This book takes all the research, practice, management experience, and managed experiences, and puts them into a coherent whole that allows you to learn how to manage software projects.

Most of my time, and most of the time spent by the people I have taught and worked with over the years, has been with the large majority of companies that are in between huge multibillion dollar companies and small shoestring-budget startups. That is, the small- to medium-size companies struggling to put process, measurement, and tools in place, and still remain financially solvent. These companies don't have instant market shares for their products and can't spend weeks and months learning a prescribed process that may or may not work for them.

The fact is that small- and medium-size companies need ways to put process, measurement, and tools in place using a stepwise approach that avoids sending managers into a panic about products not developed and drives developers to frustration with tedious overhead. This book presents a software project management approach that allows individuals, managers, and organizations to put process, measurement, and tools in place and still produce software. It also allows everyone to figure out what is going on with all three of these efforts within their organization, and how to improve them steadily over time.

Material was selected for inclusion in this book based on the 80% rule: If a subject or topic would be used by 80% of the project managers out there, it went in the book. Newer, unproven (in commercial software development where real profit means proven) methods did not get included, nor did large, documentation-heavy methods. However, the Further Information sections of each chapter point you to where you can find information on both the new and the large.

This book presents a breadth of widely used topics and focuses on how-to knowledge rather than presenting depth on a few topics. The goal of this book is not to be encyclopedic, including something about everything, but rather something about some things that you will really use. These things are topics most every software project manager will need. For example, this means that a core set of measurements are presented and the book focuses on how to use them, rather than on a broad range of sometimes-applicable software project metrics.

You will enjoy reading this book because it is not another dry computer science book. There are specific instructions as to how to accomplish critical management tasks but you will feel like you are learning them from a friend rather than from a formal instructor. There are not dozens of scripts and forms that you *have* to use, or lists of rules to follow without guidance as to how to put them together into a coherent whole. This book contains a

coherent integration of excellent information from many different sources. This integration will guide you through management of a project by using the best practices in the industry, the experience of a well-travelled professional, and common sense.

Vision

The specific purpose of this book is to teach people the basics of software project management. The approach that this book takes is much like the old saying, "Give a man a fish and you feed him for a day. Teach a man to fish and you feed him for a lifetime." Teaching a person to manage projects through mandated processes from institutes or processes sold by commercial tool vendors is like giving a person a fish: The processes have to be followed completely to be effective. Instead, this book teaches people the four basic building blocks of software project management and allows them to choose and use methods in each area that work best for their projects and organizations. This is like teaching a person to fish.

This book will help you manage software projects. If you read the book thoroughly and try the methods and approaches suggested, you will begin to gather the hands-on experience that will improve your management skills with every project. Some software project management methods and approaches are not described here and some of these may be considered "critical," "historic," or "landmark." However, there are references to other materials in the book so that an instructor or reader is free to include them to complement a chapter if needed or desired.

Prerequisites

The reader should have a basic understanding of software engineering prior to using this book. He or she should be familiar with the terms and concepts discussed, which draw from the accepted best practices in software engineering. The reader may have had a software engineering or information technology course in the past or simply the real-world experience of working in software engineering.

This text supplements and extends chapters on software project management in books by Sommerville and Pressman. This book presents more detailed, in-depth information on topics outlined in those books. Furthermore this text integrates leadership, teamwork, decision-making, and real-world project solutions with these topics. To make this integration and extension most effective, the reader should have some background knowledge of software development and maintenance.

If you are familiar with PSP or TSPi, this book will complement those processes by bringing real-world people and project issues to light. You can easily apply PSP and TSPi to the projects and exercises throughout the text. If you have not used PSP or TSPi, this book will still work very well for you. While PSP and TSPi are mentioned, the methods and appproaches described here in no way depend on or are based on either of these two processes.

Tools

This book introduces several software engineering tools. None of these tools is complex, yet all provide valuable support in estimating, organizing, monitoring, and managing software projects. CASE and COSMOS implement function point and COCOMO estimation methods. Training time on these tools involves only an hour or two. MS Project helps organize, monitor, and manage projects through schedule creation and tracking. While this tool has dozens of features, some of which are complex, this book introduces the basics and allows you to learn more features if desired.

Specific portions of this book refer to other software engineering tools that support design documentation, code metrics, documentation measures, and risk assessment. Rational tools take the forefront in some of these areas and can be further investigated through the Rational web site.

Organization

The organization of this book follows an effective learning path. Part One covers the four basic building blocks of software project management: people, process, tools, and measurements. These building blocks are stressed throughout the remaining parts of the book. From Part One, you should approach the rest of the book and every software project with these four building blocks in mind.

Part Two describes putting the four building blocks into practice to prepare for a software project. Each chapter in Part Two focuses on an area a manager needs to consider prior to project launch. Schedule and effort estimates based on classic estimation methods implemented by a simple free tool are described. Pointed discussion of the content and organization of a Software Development Plan is given.

Part Three applies the information presented in Parts One and Two to a project as it is launched and moves toward a stable state. The early phases of most projects encounter instability as requirements change and become more precise, the process is put into practice, tools are brought on line,

measurements are gathered, and, most importantly, the team gets used to each other and their individual roles. Preparing for a project is easier than managing a project through the project life cycle.

Part Four describes how to make sure that software engineering practices are put into place. As the project moves from instability to stability to delivery, there will be hundreds of product details and tasks for a project manager to monitor. The team has come to understand each other, the process, and the product, and the end of the project is now in sight. This is exactly when both you and the team need to make sure that all details are accounted for and that no unexpected problems arise that would lead to project defeat.

Part Five is critical for two reasons: (1) Undeployed or unsuccessfully deployed projects are failures; and (2) unassessed projects provide no information to you or the team that can be used to improve the next project. To work long and hard on a project and not learn from this effort is just wasteful. This book continually describes and advocates hands-on experience coupled with assessment. No silver bullet yet exists for software development but through effective measurement and assessment you can make a beginning.

Supplements

The following supplements are available to all readers of this book at www.aw.com/cssupport.

- MS PowerPoint slides
- Companion web site that provides examples of the documents, measurements and assessments from the author's commercial work
- Software estimation tools and PSP Studio Automation

The solutions to selected exercises are available only to qualified instructors. Please contact your local Addison-Wesley sales representative, or send e-mail to aw.cse@aw.com, for information about how to access them.

Parting Shot

Way back in graduate school, I was leaving the final exam of an advanced software engineering course with a good friend of mine who was visibly frustrated with the exam. Because her research area was formal language theory, I tried to comfort her by reminding her that the course was not critical to her doctoral dissertation. In frustration, she quipped, "I can do formal languages but I can't do chit-chat!" My immediate thought was "Maybe that's because 'chit-chat' isn't so dog gone easy!"

Software project management requires leadership, social, technical, personality, and motivational skills. You need skills in all these areas to solve a problem whose definition changes over time, in an environment filled with risks and unforeseen, potentially critical, events. In comparison, proving a theorem involves solving a problem that never changes, worked on by a single person, in an environment with little risk, and that poses no ramification if the problem goes unsolved. In this light, software projects don't look so soft anymore!

Acknowledgments

I would like to thank the tremendous production and marketing teams that helped make this book possible. Joyce Wells and her staff designed a terrific cover for this book. Given that I am "artistically challenged" and color blind, the cover might have been drab or hideous if left to my devices! Nancy Lombardi made my writing more precise, concise, and readable. She was very patient with me and thus improved my writing a great deal! I also want to thank Juliet Silveri for coordinating all those working to get this book together and on the street.

Maite Suarez-Rivas and Katherine Harutunian deserve huge thanks as well. They guided me through the writing and publishing process with much patience and expertise. Even when I had to stop writing temporarily owing to outside influences, they never lost confidence that I would finish.

Thanks to the reviewers who waded through multiple versions of this text: William Ayen, University of Colorado at Colorado Springs; Don Bailes, East Tennessee State University; Martin L. Barrett, East Tennessee State University; John Dalbey, California Polytechnic State University; Henry A. Etlinger, Rochester Institute of Technology; Kurt D. Fenstermacher, University of Arizona; Dennis Groth, Indiana University School of Informatics; David A. Gustafson, Kansas State University; Drew Hamilton, Auburn University; Sallie Henry, Virginia Tech; Greg Jones, Utah State University; Daniel Joyce, Villanova University; John Lewis, Villanova University; Nenad Marovac, San Diego State University; Bruce R. Maxim, University of Michigan at Dearborn; Jonathan I. Maletic, Kent State University, James M. Purtilo, University of Maryland; and Frank H. Young, Rose-Hulman Institute of Technology. To those who provided valuable input on content and guidance, thank you for help in making this book better. Special thanks to Martin Barrett and Drew Hamilton for their encouragement.

I would like to thank two mentors, Dr. Dennis Kafura and Dr. Gary Harkin, for their advice and encouragement. They continued to mentor me long after I left Virginia Tech and Montana State University.

Thanks to my parents for convincing me I could do anything I set my mind to. Thanks to my sons for giving me more credit and fame than I deserve. One son now wants to be a writer and the other drew the climber on the back cover. Thanks finally to my lovely wife for patiently watching me work long hours on too many projects. Someday I'll slow down. I promise.

About the Author

JOEL HENRY has worked in the field of software engineering for more than twenty years, mostly as a software project manager, technical lead, or software engineer. Throughout, he has concentrated on real-world issues involved in building software products for people who need them.

After receiving a PhD in Computer Science from Virginia Tech, where he focused on quantitative software process improvement, Joel moved to academia.

Always drawn to commercial software development, Joel worked before, during, and since his graduate studies on commercial software projects. These projects involved EPA air quality (stored on tape!), expert systems, Naval weapons systems, Air Force aircraft, wind tunnels, software engineering tools, aerospace vehicles, and factory control systems.

Joel has not driven a dotcom company to huge initial stock offerings but has rowed an aluminum boat 61 miles around Yellowstone Lake. Putting this same work ethic into his software engineering projects, he has put valuable software products in the hands of those who needed them on time and within budget even though they didn't bring him fame or fortune.

Joel has worked in both the academic and commercial software worlds, observing, recording, and evaluating tools, techniques, and culture in each. He prides himself on leveraging information and experience from academic and commercial environments, mixed with the wisdom of his father and grandfather, to improve both worlds.

Brief Contents

Contents

part one Understand the Basics