软件工程与方法丛书

影印版

软件同级评审

Peer Reviews in Software A Practical Guide

(美) Karl E. Wiegers 编著

Pearson Education

科学出版社
www.sciencep.com

软件工程与方法丛书

软件同级评审

（影印版）

Peer Reviews in Software:
A Practical Guide

（美） Karl E. Wiegers 编著

科学出版社

北京

图字：01-2003-7661 号

内 容 简 介

本书介绍了软件同级评审的整个过程，提供了保证软件质量的方法和技术，内容涵盖正式和非正式的评审过程、评审方法以及这些方法的适用场合。此外书中还探讨了各种影响评审计划实施的因素。本书案例丰富，简明、易懂，实用性强，适于从事软件开发和软件项目管理特别是质量管理的人员使用，也可作为高等院校研究生和本科生的软件工程类教材。

图书在版编目(CIP)数据

软件同级评审= Peer Reviews in Software: A Practical Guide /（美）维杰斯（Karl E. Wiegers）编著.—影印版. —北京：科学出版社，2004
（软件工程与方法丛书）
ISBN 7-03-012490-1
Ⅰ.软… Ⅱ.维… Ⅲ.软件开发—评估—英文 Ⅳ.TP311.52

中国版本图书馆 CIP 数据核字（2003）第 103046 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：15 1/2
印数：1—3 000 字数：240 000

定价：26.00 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。

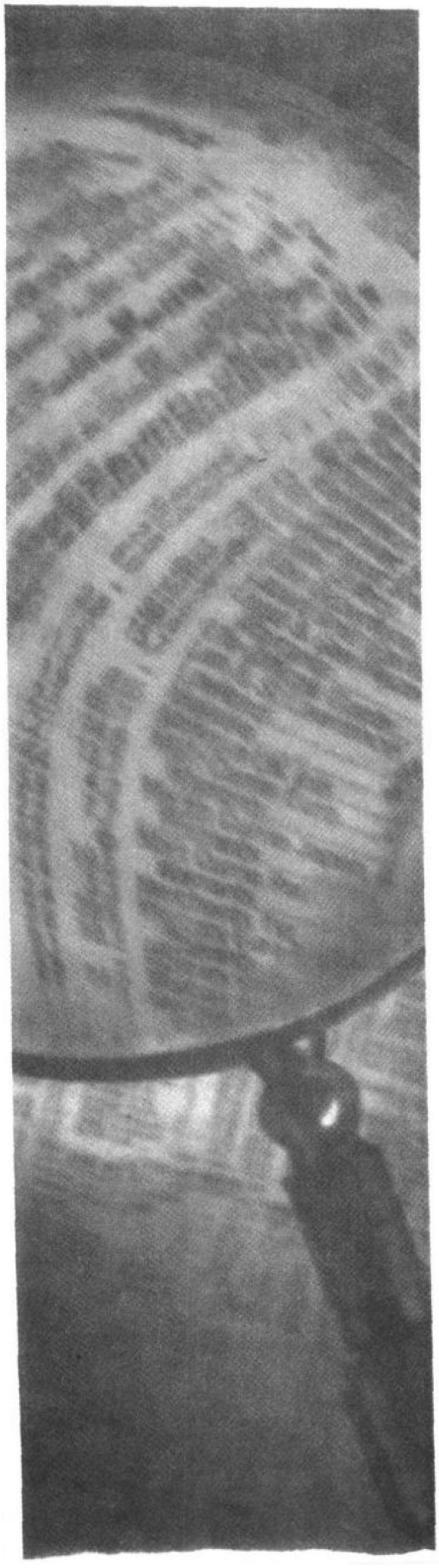
软件开发中出现错误在所难免，关键是要及早发现，而不要让其扩散，增加纠正的成本。软件同级评审制度是任何高质量软件开发过程的重要环节，然而受过这方面必要培训的专业人员却还是凤毛麟角。Karl E. Wiegers 的《软件同级评审》（*Peer Reviews in Software*）就是针对这方面需求而编写的。本书介绍了软件同级评审的全过程，内容涉及各种正规和非正规的评审方法和质量保证技巧。书中对大型项目及开发团队地域分散等情况下的同级评审进行了专门探讨。对于项目管理者而言，本书能够帮助他们以实用的资源启动同级评审计划，增进开发人员间的沟通，最终按期提交高质量的软件产品。

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



Preface

No matter how skilled or experienced I am as a software developer, requirements writer, project planner, test engineer, or book author, I'm going to make mistakes. There's nothing wrong with making mistakes; it's part of what makes me human. But because I err, it makes sense to catch the errors early, before they become difficult to find and expensive to correct.

Finding my own errors is often hard because I am too close to the work. Many years ago I learned the value of having some colleagues look over my work and point out my mistakes. I always feel a bit sheepish when they do, but I prefer to have them find the mistakes now than to have customers find them much later. Such examinations are called *peer reviews.* There are several different types of peer reviews, including inspections, walkthroughs, and others. Most of the points I make in this book apply to any activity in which someone other than the creator of a work product examines it in order to improve its quality.

I began performing software peer reviews in 1987; today I would never consider a work product complete unless someone else has carefully examined it. You might never find all of the errors, but with help from other people you will find many more than you possibly can on your own. The manuscript for this book and my previous books all underwent extensive peer review, which contributed immeasurably to their quality.

My Objectives

There is no "one true way" to conduct a peer review, so the principal goal of this book is to help you effectively perform appropriate reviews of deliverables that people in your organization create. I also address the cultural and practical aspects of implementing an effective peer review program in a software organization. Inspection is emphasized as the most formal and effective type of peer review, but I also describe several other methods that span a spectrum of formality and rigor. Many references point you to the extensive literature on software reviews and inspections.

Inspection is both one of the great success stories of software development and something of a failure. It's a grand success because it works! Since Michael Fagan developed it at IBM in the 1970s, inspection has become one of the most powerful methods available for finding software errors. You don't have to just take my word for it, either. Experiences cited from the software literature describe how inspections have improved the productivity of many software organizations and the quality of their products. However, only a fraction of the software development community understands the inspection process and even fewer people practice inspections properly and effectively. To help you implement inspections and other peer reviews in your team, this book emphasizes pragmatic approaches that any organization can apply.

Several process assets that can jumpstart your peer review program are available from the Web site that accompanies this book. Find it by going to one of these locations:

http://www.processimpact.com/pr_goodies.shtml

http://www.awl.com/cseng/

These resources include review forms, defect checklists, a sample peer review process description, spreadsheets for collecting inspection data, sources of training on inspections, and more, as described in Appendix B. You are welcome to download these documents and adapt them to meet your own needs. Please send your comments and suggestions to me at kwiegers@acm.org. Feedback on how well you were able to make peer reviews work in your team is also welcome.

Intended Audience

The material presented here will be useful to people performing many project functions, including:

- Work product authors, including analysts, designers, programmers, maintainers, test engineers, project managers, marketing staff, product managers, technical writers, and process developers
- Work product evaluators, including quality engineers, customer representatives, customer service staff, and all those listed as authors
- Process improvement leaders
- Managers of any of these individuals, who need to know how to instill peer reviews into their cultures and also should have their own deliverables reviewed

This book will help people who realize that their software product's quality falls short of their goals and those who want to tune up their current review practices, establish and maintain good communications on their projects, or ship high-quality software on schedule. Organizations that are using the Capability Maturity Model for Software or the CMMI for Systems Engineering/Software Engineering will find the book valuable, because peer reviews are components of those process improvement frameworks (see Appendix A).

The techniques described here are not limited to the deliverables and documents created on software projects. Indeed, you can apply them to technical work products from any engineering project, including design specifications, schematics, assembly instructions, and user manuals. Any business that has documented task procedures or quality control processes will find that careful peer review will discover errors the author simply cannot find on his own.

Reading Suggestions

To gain a detailed understanding of peer reviews in general and inspections in particular, you can simply read the book from front to back. The cultural and social aspects of peer reviews are discussed in Chapters 1 and 2. Chapter 3 provides an overview of several different types of reviews and suggests when each is appropriate. Chapters 4 through 8 address the nuts and bolts of inspection, while Chapter 9 describes important inspection data items and metrics. If you're attempting to implement a successful review program in an organization, focus on Chapters 10 and 11. For suggestions on ways to deal with special review challenges, such as large work products or distributed development teams, see Chapter 12. Refer to the Glossary for definitions of many terms used in the book.

Acknowledgments

What could be a more appropriate candidate for peer review than a book on peer reviews? My stalwart reviewers for this book were Sandy Browning, Tim Bueter, Rodger Drabick, Chris Fahlbusch, Lynda Fleming, Kathy Getz, Robin Goldsmith, Ellen Gottesdiener, Brian Lawrence, Karen Mattheessen, Mark Paulk, John Pustaver, Jenny Stuart, Troy Taft, Don Thresh, Scott Whitmire, Nancy Willer, and Ralph Young. They offered valuable improvement suggestions for nearly every chapter. I also appreciate those who commented on selected sections, including Brad Appleton, James Bach, Mike Dahlhausen, Zarrin Ghaemi, Tom Gilb, Bob Glass, Tammy Hoganson, Claude Laporte, Gloria Leman, Ron Radice, Phil Recchio, Kathy Rhode, Erik Simmons, and Dan Wall.

I'm grateful to Brian Lawrence for generously sharing his excellent treatise "Effective Peer Reviews" with me and to Kathy Rhode for relating her inspection experiences. Thanks also to David Gelperin for suggesting that the world needed a new book on software peer reviews and for imparting his wisdom regarding inspections.

It was a pleasure working with acquisitions editor Deborah Lafferty and production coordinator Patrick Peterson of Addison-Wesley. Malinda McCain did a fine job of editing the initial manuscript into final form, and Stratford Publishing Services rendered my simple sketches into effective figures.

And once again: thank you for your support and patience, Chris. They get easier with practice.

About the Author

Karl E. Wiegers is the Principal Consultant with Process Impact, a software process consulting and education company in Portland, Oregon. His interests include software quality engineering, requirements engineering, project management, risk management, metrics, and software process improvement. Previously, he spent eighteen years at Eastman Kodak Company, where he held positions as a photographic research scientist, software developer, software manager, and software process and quality improvement leader. Karl received a B.S. degree in chemistry from Boise State College and M.S. and Ph.D. degrees in organic chemistry from the University of Illinois. He is a member of the IEEE, IEEE Computer Society, and ACM.

Karl is the author of the books *Software Requirements* (Microsoft Press, 1999) and *Creating a Software Engineering Culture* (Dorset House Publishing, 1996), both of which won Productivity Awards from *Software Development* magazine. He has also written more than 140 articles on many aspects of computing, chemistry, and military history. Karl has served as a member of the Editorial Board for *IEEE Software* magazine and as a contributing editor for *Software Development* magazine. He is a frequent speaker at software conferences and professional society meetings. When not in front of the keyboard, Karl enjoys cooking, wine, studying military history, riding his Suzuki VX800 motorcycle, and playing his Gibson Les Paul and Fender Stratocaster guitars.

Contents

CHAPTER ONE

The Quality Challenge

"Hey, Maria, do you have a minute? I can't find a silly little bug in this program. Can you take a look at this code for me, please?"

"Sure, Phil. What's it doing wrong?"

"It's not aligning these images correctly. They're all supposed to be left-aligned, but each one is indented farther in. I'm pretty sure the problem is in the DisplayImage function, but I've been looking at it for 15 minutes and I just can't see anything wrong."

"Hmmm, let's see here. It looks like . . ." [mutter, mutter, mutter] "No, that part looks fine. But look at the top of this loop." [Maria points to the screen.] "I think this parenthesis is in the wrong place. If you move the index variable outside that paren, I don't think the images will be stairstepped any more."

Phil smacks his forehead with his palm. "You're right! Thanks a lot, Maria. I can't believe I didn't see that."

"No problem, Phil. I'm glad to help."

Most programmers have asked their colleagues to help them find elusive problems in their code. Often you are too close to your own work to spot errors you've made. As you study the code, your brain just recites what you created earlier (or what you intended to create) because you're following the same reasoning you used when you made the mistake. You need a fresh perspective—a pair of eyes that hasn't seen the code before and a brain that thinks in a different way.

Looking Over the Shoulder

In a *peer review*, someone other than the author of a work product examines that product to find defects and identify improvement opportunities. A *defect* (also known as a bug or a fault) is a condition in a software work product that would cause the software to produce an unsatisfactory or unexpected result. Phil asked Maria to conduct a short code review because he knew he was stuck, and Maria was able to get him unstuck after studying the problem code for just a few moments. Even if your organization adopts a formal peer review process, continue to rely on the kindness of your colleagues through these quick, ad hoc reviews.

People conduct various types of project reviews besides peer reviews intended to discover defects; see Table 1–1 for some examples. Although all of these reviews contribute to successful project execution and might involve "peers" of the author, this book focuses on peer reviews whose primary objective is to improve product quality.

Table 1–1. Types of Project Reviews

Review Type	*Purpose*
Educational review	Bring other stakeholders up to speed on technical topics pertinent to the project
Management, readiness, or gate review	Provide information to senior managers so they can decide to release a product, continue (or cancel) a development project, approve (or reject) a proposal, change project scope, adjust resources, or alter commitments
Peer review	Look for defects and improvement opportunities in a work product
Post-project review	Reflect on a recently completed project or phase to learn lessons for future projects
Status review	Update the project manager and other team members on progress toward milestones, problems encountered, and risks identified or controlled