软件工程与方法丛书

影印版

对象软件项目求生法则

Surviving Object-Oriented Projects

(美) Alistair Cockburn 编著

科学出版社
www.sciencep.com

软件工程与方法丛书

对象软件项目求生法则

（影印版）

Surviving Object-Oriented Projects

（美） Alistair Cockburn 编著

科学出版社

北京

图字：01-2003-7663 号

内 容 简 介

本书论述了面向对象软件开发项目中所有重要的工作内容，包括进度安排、预算、人员配置以及成本控制中可能的风险及相应的解决办法。关键内容都附有短小的真实案例，可以帮助管理人员应对对象软件项目中各种无法预知的问题，取得整个项目的成功。

本书适合从事软件开发和软件项目管理的人员使用，也可作为软件企业培训或高等院校软件工程类课程的辅助教材。

图书在版编目(CIP)数据

对象软件项目求生法则= Surviving Object-Oriented Projects /（美）考克伯恩（Alistair Cockburn）编著.—影印版. —北京：科学出版社，2004
（软件工程与方法丛书）
ISBN 7-03-012489-8
Ⅰ.对… Ⅱ.考… Ⅲ.面向对象语言—软件开发—英文 Ⅳ.TP311.52

中国版本图书馆 CIP 数据核字（2003）第 103053 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：17
印数：1—3 000 字数：260 000
定价：28.00 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

Object technology is new, powerful, intriguing, and frankly, sometimes daunting. Even if trade rags shout of object ubiquity, your programmers clamor for cool object tools, and your old ways of doing things seem stale, it still is a big, big decision. Whether objects are right for your organization and project hinges on a gritty assessment of your ability to use objects and whether introducing object technology solves more problems than it creates.

One question on the minds of veteran managers is whether their prior experience rolls over to this new object development territory. Indeed, managing objects isn't all new stuff. Object management means managing people and processes, taking measured risks, and recovering from unexpected problems. To successfully employ objects, you need a clear understanding of your people, their capabilities, their roles, your development process, and the type of project you are undertaking. It also means taking seriously this business of incremental and, most likely, iterative development. The way to acquire new skills is through practice, practice, and even more practice. A one-shot development cycle doesn't give you time to recover from your mistakes.

Alistair Cockburn sheds light on incremental and iterative development processes that would benefit any development project regardless of its technology base. But object technology, which Alistair points out as being primarily a packaging technology, comes ready equipped with tools that help build neat packages—encapsulation of data and function in objects, and reuse through composition and class inheritance.

In this book, Alistair tackles the subject of managing, staffing, and building a learning organization. In so doing, he doesn't pander to object hype. He doesn't pull any punches—he is not a fan of textbook methodologies that ignore process, upper-CASE tools that perpetually show promise but don't deliver, or overly complex languages or solutions. In fact, while letting the air out of the object-hyped balloon, he does an admirable job of stating where objects work well, and more important, how to do so without having to take an all-or-nothing plunge.

I wish you many successes on your development path, and hope you not only survive but also thrive as you embark on your journey.

Rebecca J. Wirfs-Brock
October 1997

Preface

If you are thinking of starting, or have already started, an object-oriented (OO) project, and want to know what you are up against, this book will be of use to you. Organizations that have successfully made the transition to object technology claim significant time-to-market reduction. Developers say object orientation is a fun way to develop software.

There is no shortage of literature on the subject; however, the press has made so much of object technology that it is hard to sort out exaggerations and selective reporting from the actual experience one can expect. Speakers rarely seem to want to name the actual costs of making the move to objects, perhaps to avoid scaring away future newcomers.

There is, therefore, a lack of information on what sorts of unpleasant surprises await one when starting an OO project, and what to do about them. It is this lack of information that *Surviving Object-Oriented Projects: A Manager's Guide* addresses.

SCOPE

This book covers issues I have found in several dozen organizations that are doing object-oriented projects. From failed efforts, we learned specific difficulties; from successful projects, we learned how to get around the difficulties.

The early reviewers of this book unanimously said that it takes several projects to apply the lessons. They said that people on their first OO project are not sufficiently aware of the issues to detect them, are not yet

open to suggestions, and cannot set aside old habits and thinking patterns. It is my hope that you can prove them wrong, that you will succeed on your first, or your second, project by paying attention to the lessons from other people's experiences.

This book is not a primer on object technology, nor is it a primer on object-oriented design techniques or of macro- or micromanagement. It is not a technical review of the literature, nor a cataloging of project types. It addresses the numerous, specific topics for which an answer cannot easily be found in the literature, or the obvious answer does not work.

The information in this book is based on personal project experiences—my own, those of people I have interviewed, and interviews I have read. Since writing the early version, I have had the chance to see these ideas applied to development of many kinds of systems, from non-object-oriented systems involving mainframes, COBOL and assembler, to client/server and web-based systems.

In this book, I identify topics, point out hazards, and name a workable strategy taken from a project that successfully cleared the hazard. The hazards and strategies are collected at the back of the book in Appendix B, Crib Sheets.

For readers interested in an introductory companion to this book, I recommend David Taylor's *Object Technology: A Manager's Guide, Second Edition* (Reading, MA: Addison Wesley, 1998). For extended investigation, I suggest Steve McConnell's *Rapid Application Development* (Microsoft Press, 1996).

AUDIENCE

Surviving Object-Oriented Projects: A Manager's Guide is intended for the busy professional. Here is a reading strategy for four types of possible readers:

1. *The executive scanning for impact to the organization.* Read the Preface and the first two chapters. These take you through concepts, project histories, expectations, and costs. If you are interested in the next level of depth, skim the setup issues involving project selection and staffing, and the chapters on large projects. At that point, you may wish to give the book to the project manager.
2. *The manager before starting on a project.* Read Chapters 1, 2, 3, and 5 of the book. These cover expectations, project setup, and incremental and iterative development. Then look through the list of strat-

egies (Appendix A) and hazards (in Appendix B) at the back of the book to get a sense of the total set of issues.

3. *The manager on a project.* The primary audience for this book is the manager, working with the technical lead, on a project. Some issues are technical enough to require terms of object technology. You, the project manager, may find that the technical lead will bring these problems to your attention or will help you work through them.

   Skim the entire book to get the nature and location of topics. When working on the project, look up particular topics as they occur. The chapters are organized in roughly the order topics usually show up and need to be dealt with. Reread Chapter 5 about making corrections before each increment.

4. *The project's technical leader.* Use the book to help your manager understand certain issues, such as organizing teams, developing iteratively, and resisting unproductive tools and activities. I have added technical depth in a few areas where a project hazard lurks and there is no simpler way to carry through on the discussion.

   Among these are: simplistic modeling of the business (sometimes passing under the name of "analysis"), overstaffing at the beginning of a project, and false productivity measures. For some issues, it is up to you, the technical leader, to work with the arguments in the book to convince other developers and the management team to adopt a sensible direction.

I have included an extended section on C++ because it is my carefully considered opinion that C++ represents an additional hazard to the survival of a project. If you favor using C++, read through this section and deal with the issues to ensure your project's success.

ORGANIZATION

The book has eight chapters and two appendices, roughly matching the chronology of encounter with the issues.

- Chapter 1 summarizes and introduces sucess and failure factors, and defines terms you will have to become familiar with.
- Chapter 2 is a reality check. What expectations do you have about object technology, and how should you adjust those expectations? The chapter contains stories about a dozen projects, which are referred to throughout the book.

- Chapter 3 deals with selecting and setting up a project. This is the best place to work on survival, even if survival means walking away from the project. It covers all the standard issues of staffing, training, tool selection, methodology, legacy systems, and the like.
- Chapter 4 covers some basic issues you will encounter when running the project: methodology, estimates, plans, milestones, measurements—and design.
- Chapter 5 deals with the inevitable corrections you will have to make. I start by citing my favorite project, which started dismally and then was turned around. From this project, we can learn a great deal about fine-tuning. You will have much tuning to do; do not feel bad about making changes during a project.
- Chapter 6 is written as a reflection on the first five chapters. You can pretend that you have just finished your project and are giving advice to another person who is about to start a project. What would you highlight for him or her? Here is where you can get hindsight in advance.
- Chapter 7 addresses organizations that have safely made it to the point (or declared themselves at the point) of committing large numbers of their staff to using object technology. There are new costs and new dangers lying in wait for those who move on to larger projects.
- Chapter 8 is another reality check in which I compare the contents of the book to a real project. It opens the topic of organizational culture in overall software success.
- Appendix A is a collection of 12 success strategies presented in a medical diagnosis metaphor.
- Appendix B is a summary—a "crib sheet"—to copy and tape to the wall as your daily reminder about the basics of an object-oriented project. A condensed version is printed on a card at the back of the book.

Throughout the book, the topics cross-reference each other extensively. To keep the reading uncluttered, the links to other pages are noted in the margin with a page number, as shown here.

chapter 1, 1

Your project's survival depends on developing your insights and reflexes. To help you develop them, I use material taken primarily from first- and second-hand experiences, my own and those of the many people I have interviewed. I devote space to a few published papers, which were carefully done and provide insights. They are noted in footnotes or in further reading sections in some of the chapters.

PLACE IN THE CRYSTAL BOOK COLLECTION

The *Crystal* collection highlights lightweight, human-powered development of software. Crystal works from two basic principles:

- Software development is a cooperative game of group invention and communication. Software development improves as we improve people's personal skills and improve the team's collaboration effectiveness.
- Different projects have different needs. Systems have different characteristics, and are built by teams of differing sizes, containing people having differing values and priorities. It cannot be possible to describe the one, best way of producing software.

The foundation book for the collection is *Software Development as Cooperative Game*. It works out the ideas of software development as a cooperative game, of methodology as a coordination culture, and of methodology families (categorized in colors: clear, yellow, orange, red, and so on). It separates different aspects of methodologies: roles, techniques, activities, work products, standards, and so on.

Some books discuss a single technique or role on the project, and some discuss team issues. *Surviving Object-Oriented Projects* is one of the latter. It presents some of the projects that helped me see the primacy of people in developing software, and includes ideas for working with people on a team. It contains a brief description of a *Crystal/Orange* methodology, aimed at IT projects of 25-50 people. It goes through the issues that affect many projects, issues ranging from selection of technology to staffing to scheduling strategies. As with all the *Crystal* collection, this book is self-contained in the topics it addresses.

Acknowledgments

I thank the nontechnical people around me: Kieran, Sean, Cameron, Deanna. I now know why so many authors thank their families. Thanks also to the people at Beans & Brew, who provided good coffee, a good atmosphere, and good conversation.

As Plato said:

> Only if the various principles—names, definitions, intimations and perceptions—are laboriously tested and rubbed one against the other in a reconciliatory tone, without ill will during the discussion, only then will insight and reason radiate forth in each case, and achieve what is for man the highest possible force. . . .

This book received much benefit from the testing of principles and the "rubbing together" of ideas in conciliatory tone and without ill will. For that I thank the following individuals:

- Bruce Anderson, IBM Object Technology Practice
- Carol Burt, President, 2AB, Inc.
- Dave Collins, Outback Software, Ltd.
- Anton Eliens, Vrije Universiteit (Amsterdam)
- Adele Goldberg, Neometron
- David Gotterbarn, East Tennessee State University

[1]From Plato's 7th letter, cited in translation from Göranzon, B., Florin, M., *Dialogue and Technology: Art and Knowledge*, p. 46 (London: Springer-Verlag), 1991.

- Brian Henderson-Sellers, Swinburne University of Technology
- Jeremy Raw, Independent Consultant (Durham, NC)
- Arthur J. Riel, Vangaurd Training, Inc.
- Cecilia Shuster, Independent Consultant
- Dave Thomas, IBM (formerly of OTI)
- Daniel Tkach, IBM Consulting Group
- Rebecca Wirfs-Brock, Wirfs-Brock Associates
- The editors at Addison-Wesley

for helping to improve the book. I am indebted to Sam Adams for the terms *big-M* and *little-m* methodology, and to Dick Antalek and Wayne Stevens who taught me the most about big-M methodologies. Marilyn Rash at Addison Wesley Longman helped save my sanity during the production of this book.

I also want to thank the authors of the Eyewitness Accounts for taking time to contribute their knowledge about object-oriented projects:

- Jim Coplien, Bell Labs
- Ward Cunningham, Cunningham & Cunningham
- C.D., Independent Consultant
- Harvie S. (Sam) Griffith, Jr., President, Object Methods Software
- Luke Hohmann, SmartPatents
- Glenn House, formerly of Mentor Graphics
- K.L., Independent Consultant
- Jon Marshall, ParcPlace-Digitalk
- Tom Morgan, Brooklyn Union Gas
- Jeremy Raw, Independent Consultant

K.L. and C.D. asked that I not use their names, in order not to discomfit any companies.

All these people were kind enough to contribute their experience to this book. They may not agree with everything I write, but we all share the wish to help you succeed on your project.

Contents