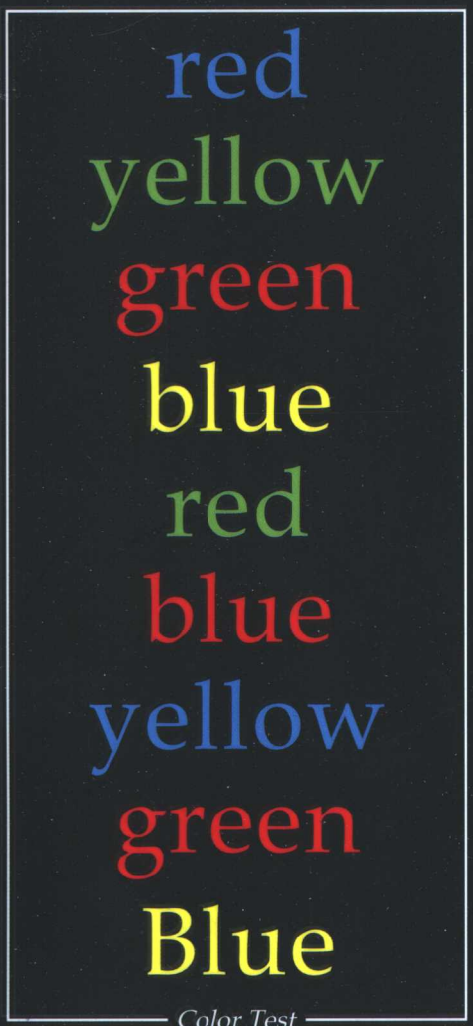
PEARSON Prentice Hall

敏捷软件开发

——使用Scrum过程（影印版）

Agile Software Development with Scrum



Color Test

[美] Ken Schwaber，Mike Beedle 著

Pearson Education 清华大学出版社

图书在版编目(CIP)数据

敏捷软件开发：使用 Scrum 过程/(美)施瓦伯(Schwaber, K.), (美)比奕(Beedle, M.)著. —影印本. —北京：清华大学出版社，2004.4
书名原文：Agile Software Development with Scrum
ISBN 7-302-08154-9

I. 敏… II. ①施… ②比… III. 软件开发—英文 IV. TP311.52

中国版本图书馆 CIP 数据核字（2004）第 013348 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社总机：010-6277 0175　　**客户服务**：010-6277 6969
责任编辑：郭福生　常晓波
封面设计：立日新
印 刷 者：清华大学印刷厂
装 订 者：三河市李旗庄少明装订厂
发 行 者：新华书店总店北京发行所
开　　本：160×230　**印 张**：10.75
版　　次：2004 年 4 月第 1 版　2004 年 4 月第 1 次印刷
书　　号：ISBN 7-302-08154-9/TP・5892
印　　数：1~3000
定　　价：25.00 元

中 文 序

“工作可以是，而且也应该是一种高尚的体验。”那么，就从本书开始你的这种体验吧，这是关于敏捷软件过程的最睿智、最实用的技术参考书之一。

软件过程是这 10 年来最热门的话题之一。我们已经知道的软件过程就有 XP、Adaptive、Crystal Clear 以及 RUP 等等。我们看到，敏捷联盟(Agile Alliance)已经成立；这是一群致力于推广可以顺利发挥作用的、面向人员的软件过程的专家。我们也看到了仅仅基于过程开发的商业软件产品。我们还看到了无数的、吹捧这种或那种过程的优点的书籍、演讲、课程和文章。

在这种喧闹声中，Ken Schwaber 和 Mike Beedle 为我们带来了 Scrum 过程。Scrum 是一种经过实践检验的敏捷软件开发方法。通过本书，你可以了解到这种方法是如何形成的；可以阅读到采用这种方法的项目的故事；可以了解作者是如何开发这种能够帮助他们在需求快速多变的情况下完成项目的方法的；可以了解到在什么情况下适合采用这种方法，在什么情况下不适合采用这种方法；他们遇到了什么问题，又是如何解决这些问题的；你还将了解到，如何使他们的方法适合自己的工作需要。

作为本书的作者，没有比 Mike 和 Ken 更合适的人选了。近 20 年来，他们一直是活跃在软件行业的专家。Mike 曾经担任许多软件项目的经理，目前经营一家非常成功的软件咨询公司。Mike 参加了许多次过程战役，他知道什么行，也知道什么不行。Ken 职业生涯的大部分时间都与软件过程有关，他定义并开发了一种软件产品，这种产品可以自动完成一些绝顶聪明的软件过程；他还创建了用于自动化软件开发的方法。Ken 知道，根据经验，这些软件过程未必能够经得起市场的考验。然而，这只是你将在本书中看到的一个故事。Ken 是一位著名的管理顾问，他已经帮助数十个项目团队成功地实现了其目标。

本书是为高级经理、软件经理、项目主管以及程序员编写的。无论你属于哪一种角色，本书都可以明确地告诉你，如何应用 Scrum 过程的简单而行之有效的原则和方法。

如果你必须完成一个项目，并且想采用一种需要时可以帮助你、不需要时不会妨碍你的过程，那么你应该读一读本书。本书很可能就是你享受高尚工作体验的一种催化剂。

Robert C. Martin

Foreword

"Work can and should be an ennobling experience." So begins *Scrum - Agile Software Development*, one of the sanest and most practical books on agile software processes.

Software process is one of the hot topics of this decade. We've seen processes like XP, Adaptive, Crystal Clear, RUP, etc. We've seen the formation of the Agile Alliance; a group of experts dedicated to the promotion of people-oriented software processes that work without getting in the way. We've seen the creation of a commercial product based upon nothing but process. And we've seen dozens, if not hundreds, of books, lectures, classes, and articles extolling the virtues of one process or another.

In the midst of this hubbub, Ken Schwaber and Mike Beedle bring us Scrum. Scrum is an agile software development method with a proven track record. In this book you will read how the method was created, and some stories of the projects that made use of it. You'll read about how the authors battled to create a method that helped them get projects done in the presence of rapidly changing requirements. You'll read about what worked and what didn't, the problems they had, and the way they solved them. You'll read about how you can adapt their work to your particular needs.

Mike and Ken are uniquely qualified to author this book. Both have been active in the software industry for decades. Mike has been a manager of many software projects, and runs a successful software consultancy. Mike has fought the process battles many times. He knows what works and what doesn't. Ken has been involved with software process for a large portion of his career. He defined and built a software product that automated heavyweight software processes and created the methodology automation industry. From this experience he learned that such processes were not amenable to creating software in real market environments. But that's a story you can read in the book. Ken is a well-known management consultant who has helped dozens of project teams reach their goals.

This is a book for executives, software managers, project leaders, and programmers. It describes, in no uncertain terms, how each of these roles can apply the simple but effective principles and techniques of Scrum.

If you have to get a project done, and you want to use a process that helps you when you need help, and gets out of the way when you don't, then you should read this book. It is liable to be the catalyst of an ennobling experience.

Robert C. Martin

Foreword

When I finished at my grammar school at 18 I spent a year working in industry before going to University. My career direction at the time was electrical engineering, and in my year I learned a great deal about the engineering approach to building things. When I left university and entered the world of software development I was attracted to graphical modeling methodologies, because they helped put engineering discipline into software development.

At the heart of the engineering approach is a separation of design and construction, where construction is the larger part of the job and is a predictable process. Over time I began to find that this separation wasn't really useful for my software work. Doing the separation required too many tasks that didn't seem to really contribute to producing software. Furthermore, the construction part of the task wasn't really that predictable, and the design portion was much longer than the engineering approach assumed.

In Chapter 2 Ken describes a particular moment that brought this question home for him, when he spend time with DuPont's process engineering experts. There he learned the difference between defined and empirical processes, and realized that his software development needed to be controlled using an empirical approach.

We aren't the only ones who've been asking these questions about the nature of software development. Over the last few years there's been increasing activity in the area of what is now called Agile Methodologies, a new breed of software processes which are based on an empirical approach to controlling a project.

And software projects do need to be controlled. For many people, moving away from defined processes means descending into chaos. What Ken learned at DuPont was that a process can still be controlled even if it can't be defined. What Ken and Mike have written here is a book that shows you one way of doing that. Practices such as sprints, scrum meetings, and backlogs are techniques that many people using Scrum have used to control projects in chaotic circumstances.

In the future, we'll see more need for Scrum and the future developments built upon it. Software development has always been difficult to control. Recent studies indicate that the average project takes twice as long to do as its initial plans. At the heart of Scrum is the notion that if you try to control an empirical process with a system designed for defined processes, you are doomed to fail. It's becoming increasingly apparent that a large proportion of software projects are empirical in nature and thus need

a process like Scrum. If you're running a project, or buying software, with difficult and uncertain requirements in a changing business world, these are the kinds of techniques you need.

Martin Fowler

Preface

This book was written for several audiences. Our first audience is application development managers that need to deliver software to production in short development cycles while mitigating the inherent risks of software development. Our second audience is the software development community at large. To them, this book sends a profound message: *Scrum represents a new, more accurate way of doing software development that is based on the assumption that software is a new product every time that it is written or composed.* Once this assumption is understood and accepted, it is easy to arrive at the conclusion that software requires a great deal of research and creativity, and that therefore it is better served by a new set of practices that generate a self-organizing structure while simultaneously reducing risk and uncertainty.

Finally, we have also written this book for a general audience that includes everyone involved in a project where there is constant change and unpredictable events. For this audience Scrum provides a general-purpose project management system that delivers, while it thrives on change and adapts to unpredictable events.

Software as "new product" as presented in this book, is radically different from software as "manufactured product", the standard model made for software development throughout the last 20 years. Manufacture-like software methods assume that predictability comes from defined and repeatable processes, organizations, and development roles; while Scrum assumes the process, the organization, and the development roles are emergent but statistically predictable, and that they arise from applying simple practices, patterns, and rules. Scrum is in fact much more predictable and effective than manufacturing-like processes, because when the Scrum practices, patterns and rules are applied diligently, the outcome is always: 1) higher productivity, 2) higher adaptability, 3) less risk and uncertainty, and 4) greater human comfort.

The case studies we provide in this book will show that Scrum doesn't provide marginal productivity gains like process improvements that yield 5-25% efficiencies. When we say Scrum provides higher productivity, we often mean several orders of magnitude higher i.e. several 100 percents higher. When we say higher adaptability we mean coping with radical change. In some case studies, we present cases where software projects morphed from simple applications in a single domain to complex applications across multiple domains: Scrum still managed while providing greater human comfort to everyone involved. Finally, we show through case studies that Scrum reduces risk and uncertainty by making everything visible *early and often* to all the people involved and by allowing adjustments to be made *as early as possible.*

Throughout this book we provide 3 basic things: 1) an understanding of why this new thinking of software as new product development is necessary, 2) a thorough description of the Scrum practices that match this new way of thinking with plenty of examples, and 3) a large amount of end-to-end case studies that show how a wide range of people and projects have been successful using Scrum for the last 6 years.

This last point is our most compelling argument: The success of Scrum is overwhelming. Scrum has produced by now billions of dollars in operating software in domains as varied as finance, trading, banking, telecommunications, benefits management, healthcare, insurance, e-commerce, manufacturing and even scientific environments.

It is our hope that you, the reader of this book, will also enjoy the benefits of Scrum, whether as a development staff member wishing to work in a more predictable, more comforting, and higher producing environment, or as a manager desiring to finally bring certainty to software development in your organization.

Thanks to our reviewers: Martin Fowler, Jim Highsmith, Kent Beck, Grant Heck, Jeff Sutherland, Alan Buffington, Brian Marick, Gary Pollice, and Tony D'Andrea. Ken would like to thank Chris, Carey, and Valerie. Mike would like to thank Laura, David, Daniel, and Sara. Together, we would like to thank our editors at Prentice Hall, Alan Apt and Robert Martin, as well as Jeff Sutherland for his many contributions to Scrum, and Kent Beck for demanding that we write this book.

Mike Beedle, Chicago
Ken Schwaber, Boston

Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

> "In today's fast-paced, fiercely competitive world of commercial new product development, speed and flexibility are essential. Companies are increasingly realizing that the old, sequential approach to developing new products simply won't get the job done. Instead, companies in Japan and the United States are using a holistic method; as in rugby, the ball gets passed within the team as it moves as a unit up the field." (Reprinted by permission of *Harvard Business Review* From: "The New New Product Development Game" by Hirotaka Takeuchi and Ikujiro Nonaka, January, 1986. Copyright 1986 by the Harvard Business School Publishing Corporation, all rights reserved.) —

This book presents a radically different approach to managing the systems development process. Scrum implements an empirical approach based in process control theory. The empirical approach reintroduces flexibility, adaptability, and productivity into systems development. We say "reintroduces" because much has been lost over the past twenty years.

This is a practical book that describes the experience we have had using Scrum to build systems. In this book, we use case studies to give you a feel for Scrum-based projects and management. We then lay out the underlying practices for your use in projects.

Chapters 5 and 6 of this book tell why Scrum works. The purpose of these chapters is to put an end to the ungrounded and contentious discussion regarding how best to build systems. Industrial process control theory is a proven body of knowledge that describes why Scrum works and other approaches are difficult and finally untenable. These chapters describe what process control theory has to say about systems development, and how Scrum arose from this discipline and theory. These chapters also lay out a terminology and framework from which empirical and adaptive approaches to systems development can ascend and flourish.

Scrum [Takeuchi and Nonaka], is a term that describes a type of product development process initially used in Japan. First used to describe hyper-productive development in 1987 by Ikujiro Nonaka and Hirotaka Takeuchi, Scrum refers to the strategy used in rugby for getting an out-of-play ball back into play. The name Scrum stuck because of the similarities between the game of rugby and the type of product development proscribed by Scrum. Both are adaptive, quick, self-organizing, and have few rests.

Building systems is hard and getting harder. Many projects are cancelled and more fail to deliver expected business value. Statistically,

the information technology industry hasn't improved much despite efforts to make it more reliable and predictable. Several studies have found that about two-thirds of all projects substantially overrun their estimates [McConnell].

We find the complexity and urgency of requirements coupled with the rawness and instability of technology to be daunting. Highly motivated teams of highly skilled developers sometimes succeed, but where do you find them? If you are looking for a quick, direct way to resuscitate a troubled project, or if you are looking for a cost-effective way to succeed with new projects, try Scrum. Scrum can be started on just one project and will dramatically improve the project's probability of success.

Scrum is a management and control process that cuts through complexity to focus on building software that meets business needs. Scrum is superimposed on top of and wraps existing engineering practices, development methodologies, or standards. Scrum has been used to wrap Extreme Programming. Management and teams are able to get their hands around the requirements and technologies, never let go, and deliver working software. Scrum starts producing working functionality within one month.

Scrum deals primarily at the level of the team. It enables people to work together effectively, and by doing so, it enables them to produce complex, sophisticated products. Scrum is a kind of social engineering aiming to achieve the fulfillment of all involved by fostering cooperation. Cooperation emerges as teams self-organize in incubators nurtured by management. Using Scrum, teams develop products incrementally and empirically. Teams are guided by their knowledge and experience, rather than by formally defined project plans. In almost every instance in which Scrum has been applied, exponential productivity gains have been realized.

As authors of Scrum, we have evolved and used Scrum as an effective alternative to traditional methodologies and processes. We've written this book to help you understand our thinking, share our experiences, and repeat the success within their own organizations.

In this book, we'll be using the word "I" from now on rather than "we", "Mike", or "Ken". Unless otherwise identified, "I" will hereafter refer to Mike Beedle in chapters 6 and 7, and to Ken Schwaber elsewhere.

1.1 Scrum At Work

The best way to begin to understand Scrum is to see it at work. After using Scrum to build commercial software products, I used Scrum to help other organizations build systems. The first organization where Scrum was tested and refined was Individual, Inc. in 1996.

Individual, Inc. was in trouble and its leaders hoped that Scrum could help them out. Individual, Inc. published an online news service called NewsPage. NewsPage was initially built using proprietary technol-

ogy and was subsequently licensed to companies. With the advent of the Internet, Individual, Inc. began publishing Personal NewsPage as a website for individuals.

Eight highly skilled engineers constituted the Personal NewsPage (PNP) product development team. Though the team was among the best I've worked with, it suffered from a poor reputation within Individual, Inc. It was said the PNP team couldn't produce anything, that it was a "total disaster." This belief stemmed from the fact that there hadn't been a new PNP release in nearly nine months. This was in 1996, when Internet time hadn't yet taken hold of the industry, but nine months was already far too long. When I discussed this situation with marketing, product management, and sales, they said they couldn't understand the problem. They would tell the PNP team what they wanted in no uncertain terms, but the functionality and features they requested never were delivered. When I discussed the situation with the disgruntled PNP team, it felt that it was never left alone to develop code. The engineers used the phrase "fire drill." The team would think about how to deliver a required piece of functionality, start working on it, and it would suddenly be yanked off onto the next hot idea. Whenever the PNP team committed to a project, it didn't have enough time to focus its attention before product management changed its mind, marketing told it to do something else, or sales got a great idea that had to be implemented immediately.

The situation was intolerable. Everyone was frustrated and at odds with each other. Competition was appearing on the horizon. I asked Rusty, the head of product management, to come up with a list of everything that people thought should be in PNP. He already had a list of his own and was reluctant to go to everyone and ask for his or her input. As he said, "If the PNP team can't even build what we're asking it to do now, why should we waste the effort to go through list building again?" However, Rusty did as I asked and compiled a comprehensive list. He also met with the PNP team to see if it knew of technology changes that needed to be made to implement the requirements. These were added to the list. He then prioritized the list. The PNP team gave development time estimates. Rusty sometimes changed priorities when it became apparent that items with major market impact didn't take much effort, or when it became apparent that items with minor market impact would take much more effort than they were worth.

I asked Rusty to change the product requirements process. People currently went straight to the PNP team to ask for new product features and functionality. I thought it could be more productive if it only had one source of work and wasn't interrupted. To implement this, Rusty suggested that people take their requests only to him. He added their requests to his list. He then reprioritized the list based on their presentation of the feature's importance, his estimate (after talking to someone on the PNP