Microsoft

Agile Project Management with Scrum

Ken Schwaber

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Cataloging-in-Publication Data
Schwaber, Ken.
Agile Project Management with Scrum / Ken Schwaber.
p. cm.
Includes index.
ISBN 0-7356-1993-X
1. Computer software--Development. 2. Project management. 3. Scrum (Computer software development) I. Title.

QA76.76.D47S32 2003
005.1--dc22 2003065178

Printed and bound in the United States of America.

5 6 7 8 9 QWE 8 7 6

Distributed in Canada by H.B. Fenn and Company Ltd.

A CIP catalogue record for this book is available from the British Library.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office or contact Microsoft Press International directly at fax (425) 936-7329. Visit our Web site at www.microsoft.com/mspress. Send comments to *mspinput@microsoft.com*.

Acquisitions Editors: Linda Engelman and Robin Van Steenburgh
Project Editor: Kathleen Atkins
Indexer: Bill Meyers

Body Part No. X10-25679

Dedicated to ScrumMasters

Contents

Foreword

My new boss wasn't being a jerk, but it seemed like it at the time. We were writing new software for use in the company's high-volume call centers. Instead of the 12 months I told him we'd probably need, he had agreed to give me 4 months. We wouldn't necessarily start using the new software in 4 months, but from that point on, all my boss could give me was 30 days' notice of a go-live date. After the first 4 months, I would have to keep the software within 30 days of releasable. My boss understood that not all functionality would be there after 4 months. He just wanted as much as he could get, as fast as he could get it. I needed to find a process that would let us do this. I scoured everything I could find on software development processes, which led me to Scrum and to Ken Schwaber's early writings on it.

In the years since my first Scrum project, I have used Scrum on commercial products, software for internal use, consulting projects, projects with ISO 9001 requirements, and others. Each of these projects was unique, but what they had in common was urgency and criticality. Scrum excels on urgent projects that are critical to an organization. Scrum excels when requirements are unknown, unknowable, or changing. Scrum excels by helping teams excel.

In this book, Ken Schwaber correctly points out that Scrum is hard. It's not hard because of the things you do; it's hard because of the things you don't do. If you're a project manager, you might find some of your conventional tools missing. There are no Gantt charts in Scrum, there's no time reporting, and you don't assign tasks to programmers. Instead you'll learn the few simple rules of Scrum and how to use its frequent inspect-and-adapt cycles to create more valuable software faster.

Ken was there at the beginning of Scrum. Ken, along with Jeff Sutherland, was the original creator of Scrum and has always been its most vocal proponent. In this book, we get to read about many of the Scrum projects Ken has participated in. Ken is a frequent and popular speaker at industry conferences, and if you've ever heard him speak, you know he doesn't pull any punches. This book is the same way: Ken presents both the successes and the failures of past Scrum projects. His goal is to teach us how to make our projects successful, and so he presents examples we can emulate and counterexamples for us to avoid.

This book clearly reflects Ken's experience mentoring Scrum Teams and teaching Certified ScrumMaster courses around the world. Through the many stories in this book, Ken shares with us dozens of the lessons he's learned. This book is an excellent guide for anyone looking to improve how he or she delivers software, and I recommend it highly.

—Mike Cohn
Certified ScrumMaster
Director, Agile Alliance

Foreword: Why Scrum Works

Suppose I'm traveling from Chicago to Boston by airplane. Before and during the flight, the pilot gets instructions from air traffic control. We take off on command and follow the prescribed route. Once we are in the air, computers predict almost to the minute when we will land in Boston. If things change—say the air is bumpy—the pilot must get permission to move to a different altitude. As we approach the airport, the pilot is told what runway to land on and what gate to go to.

If, however, I set out for Boston in a car, I can take whatever route I want, whenever I want. I don't know exactly when I'll get there, and I probably haven't planned what route I'll take or where I'll stop for the night. En route, I follow traffic laws and conventions: I stop at red lights, merge into traffic according to the prevailing customs, and keep my speed consistent with the flow. In an automobile, I am an independent agent, making decisions in my own best interests framed by the rules of the game of driving.

It's amazing to me that thousands upon thousands of people travel by car every day, accomplishing their goals in a framework of simple traffic rules, with no central control or dispatching service. It also amazes me that when I want to ship a package, I can enter a pickup request on the shipper's Web site and a driver will arrive at my door before the time that I specify. The driver isn't dispatched to each house; he or she receives a continually updated list of addresses and deadlines. It's the driver's job to plot a route to get all the packages picked up on time.

As complexity increases, central control and dispatching systems break down. Some might try valiantly to make the control system work by applying more rigor, and indeed that works for a while. But the people who prevail are those who figure out how to change to a system of independent agents operating under an appropriate set of rules. It might work to provide same-day delivery with a dispatch system that plans a driver's route at the beginning of the day. However, it is far more difficult to preplan a pickup route when customers can enter pickup requests at any time. Taxi companies sort things out at a central control center. Some shipping companies send the request to the driver responsible for the area and let the driver determine the best route based on current conditions and other demands.

The more complex the system, the more likely it is that central control systems will break down. This is the reason companies decentralize and

governments deregulate—relinquishing control to independent agents is a time-honored approach to dealing with complexity. Scrum travels this well-trodden path by moving control from a central scheduling and dispatching authority to the individual teams doing the work. The more complex the project, the more necessary it becomes to delegate decision making to independent agents who are close to the work.

Another reason that Scrum works is that it dramatically shortens the feedback loop between customer and developer, between wish list and implementation, and between investment and return on investment. Again, complexity plays a role here. When a system is simple, it's not so hard to know in advance what to do. But when we are dealing with a market economy that changes all the time and with technology that won't stand still, learning through short cycles of discovery is the tried-and-true problem-solving approach.

We already know this. We try out various marketing campaigns and discover which approach works. We simulate vehicle behavior during car design to discover the best slope of the hood and best distribution of weight. Virtually all process-improvement programs use some version of the Deming cycle to study a problem, experiment with a solution, measure the results, and adopt proven improvements. We call this fact-based decision making, and we know that it works a lot better than front-end-loaded predictive approaches.

Scrum is built on 30-day learning cycles that prove complete business concepts. If we already know everything and have nothing to discover, perhaps we don't need to use Scrum. If we need to learn, however, Scrum's insistence on delivering complete increments of business value helps us learn rapidly and completely. One of the reasons complete increments are important is that partial answers often fool us into thinking that an approach will work, when in reality, the approach doesn't work upon closer examination. We know that until software is tested, integrated, and released to production, we can't really be sure that it will deliver the intended business value. Scrum forces us to test and integrate our experiments and encourages us to release them to production, so that we have a complete learning cycle every 30 days.

Scrum doesn't focus on delivering just any increment of business value; it focuses on delivering the highest priority business value as defined by the customer (Product Owner). The Product Owner and the Team confer about what that definition is, and then the Team decides what it can do in 30 days to deliver high-priority business value. Thus the short feedback loop becomes a business feedback loop—Scrum tests early and often whether the system being developed will deliver value and exactly what that value will look like. This allows the system to be molded over time to deliver value as it is currently understood, even as it helps to develop a better understanding of that value.

Another reason Scrum works is that it unleashes the brainpower of many minds on a problem. We know that when things go wrong, there are people around who knew there was a problem, but somehow their ideas were overlooked. For example, when the space shuttle disintegrated on reentry, a widely reported interpretation of the causes of the disaster suggests that there were engineers who were well aware that there could be a problem, but they were unable to get their concerns taken seriously. What management system can we use to leverage the experience, ideas, and concerns of the people closest to the work to be done?

According to Gary Convis, president of Toyota Motor Manufacturing Kentucky, the role of managers in a healthy, thriving, work environment is "to shape the organization not through the power of will or dictate, but rather through example, through coaching and through understanding and helping others to achieve their goals."[1]

Scrum turns small teams into managers of their own fate. We know that when we are responsible for choosing our own driving route to Boston, we will find a way to get there. We will detour around construction and avoid rush hour traffic jams, making decisions on the fly, adapting to the independent decisions of all of the other drivers out there. Similarly, Scrum Teams accept a challenge and then figure out how to meet that challenge, detouring around roadblocks in creative ways that could not be planned by a central control and dispatching center.

If teams are of a size that encourages every member to participate, and team members feel like they are in control of their own destiny, the experience, ideas, and concerns of individual members will be leveraged, not squelched. When team members share a common purpose that everyone believes in, they will figure out how to achieve it. When teams understand and commit to delivering business value for their customers, when they are free to figure out how to perform tasks, and when they are given the resources they need, they will succeed.

Gary Convis notes that Toyota's sustainable success comes from an "interlocking set of three underlying elements: the philosophical underpinnings, the managerial culture and the technical tools. The philosophical underpinnings include a joint [worker], customer-first focus, an emphasis on people first, a commitment to continuous improvement.... The managerial culture...is rooted in several factors, including developing and sustaining a sense of trust, a commitment to involving those affected by first, teamwork, equal and fair treatment for all, and finally, fact-based decision making and long-term thinking."[2]

---

1. Gary Convis, "Role of Management in a Lean Manufacturing Environment," in "Learning to Think Lean,"August 2001, SAE International, *http://www.sae.org/topics/leanjul01.htm*.

2. Ibid.

Scrum works for all the same reasons. Its philosophical underpinnings focus on empowering the development team and satisfying customers. Its managerial culture is rooted in helping others achieve their goals. Its technical tools are focused on making fact-based decisions through a learning process. When all of these factors are in place, it's hard for Scrum not to succeed.

—Mary Poppendieck
Poppendieck.LLC

Acknowledgments

Special thanks to my daughter, Carey Schwaber, whose editing turns words into streams, and to Mike Cohn and Mary Poppendieck, for their fine help in keeping this book focused.

Introduction

I offer you Scrum, a most perplexing and paradoxical process for managing complex projects. On one hand, Scrum is disarmingly simple. The process, its practices, its artifacts, and its rules are few, straightforward, and easy to learn. In 2001, Mike Beedle and I wrote a short, straightforward book describing Scrum: *Agile Software Development with Scrum* (Prentice Hall). On the other hand, Scrum's simplicity can be deceptive. Scrum is not a prescriptive process; it doesn't describe what to do in every circumstance. Scrum is used for complex work in which it is impossible to predict everything that will occur. Accordingly, Scrum simply offers a framework and set of practices that keep everything visible. This allows Scrum's practitioners to know exactly what's going on and to make on-the-spot adjustments to keep the project moving toward the desired goals.

Common sense is a combination of experience, training, humility, wit, and intelligence. People employing Scrum apply common sense every time they find the work is veering off the path leading to the desired results. Yet most of us are so used to using prescriptive processes—those that say "do this, then do that, and then do this"—that we have learned to disregard our common sense and instead await instructions.

I wrote this book to help people understand how to use Scrum as they work on complex problems. Instead of further describing the framework and practices of Scrum, I offer a number of case studies in which people use Scrum to solve complex problems and perform complex work. In some of these case studies, people use Scrum correctly and the project in question ends up achieving their goals. In other case studies, people struggle with Scrum and their projects are less successful. These are people to whom Scrum is not intuitive. I've worked to understand how this can be possible. After all, Scrum is a very simple process for managing complex projects. Compared to many traditional approaches to project management, Scrum is almost effortless. Or at least I used to think it was.

Most people responsible for managing projects have been taught a deterministic approach to project management that uses detailed plans, Gantt charts, and work schedules. Scrum is the exact opposite. Unlike these tools, which practically fight against a project's natural momentum, Scrum shows

management how to guide a project along its optimal course, which unfolds as the project proceeds. I've heard that traveling along a learning curve starts from a point where you have to think everything through step by step and ends at a point where you can perform the work in question unconsciously. This is particularly true of Scrum because those steeped in traditional management practices have to unlearn many of them.

I recently helped a software development company adopt Scrum. Initially, the company had planned for two releases over the next 12 months. Because of its success in using Scrum, however, most of the functionality from the two releases was ready within 5 months. But when I visited the engineering organization, the staff was working weekends and nights to put even more functionality into the release. Even though the engineers had been wildly successful, marketing still was berating them for not delivering enough and living up to "commitments." The engineers were feeling guilty for not doing everything that marketing said was necessary, and they were ruining their personal lives to try to do everything marketing requested. This pathology had persisted despite the fact that the engineers had already accomplished the work involved in two releases in the time usually allotted for one. Old habits die hard.

Another change that Scrum engenders can best be described by thinking of how a house is built. The buyer of the house cannot move into the house until the entire house is completed. Suppose that there were an incremental, iterative approach for home construction. Suppose that using this approach, houses were built room by room. The plumbing, electrical, and infrastructure would be built in the first room and then extended to each room as it was constructed. Buyers could move in as soon as they had decided that enough rooms had been completed. Then additional rooms could be constructed depending on the needs of the buyer. Scrum lets buyers have software built in this fashion. While the infrastructure is deployed, pieces of functionality are delivered to buyers so that their organizations can start using parts of the system early in the development cycle. As the system is experienced, the buyer can determine which parts of the system will be constructed in what order and use these parts as they are completed. Buyers might even choose not to have the entire system built if they are satisfied with only a subset of the total functionality they'd initially envisioned.

I used to teach people the theory, practices, and rules of Scrum. Now I teach them what Scrum feels like as it is implemented. I teach them how to recognize when things are going right and when they are going wrong. I provide exercises and discussions that let them experience the epiphanies so that they know what Scrum should feel like. Just as you don't really know what it's like to be someone else until you've walked however many miles in his or her

shoes, you might not fully understand Scrum until you implement it yourself. But as you read this book, you will begin to understand what Scrum feels like and how you might feel using Scrum in your organization.

How should you read this book, which is in essence a book of case studies about Scrum? I've provided some of the background for each story, described how Scrum was used in that situation, and presented some of the lessons that can be learned from the way Scrum was used. The case studies are organized into topical chapters, through which you should feel free to browse. The chapter topics are Chapter 1, "Backdrop: The Science of Scrum; Chapter 2, "New Management Responsibilities"; Chapter 3, "The ScrumMaster"; Chapter 4, "Bringing Order from Chaos"; Chapter 5, "The Product Owner"; Chapter 6, "Planning a Scrum Project"; Chapter 7, "Project Reporting"; Chapter 8, "The Team"; and Chapter 9, "Scaling Projects Using Scrum." Sometimes I indicate that the background for a story has been provided in a previous chapter.

Appendix A, "Rules," lists the rules that are used in various Scrum practices and meetings. These rules hold Scrum together. If you are familiar with Scrum but you come across terms that you do not fully understand, you should look them up in Appendix B, "Definitions." If you are unfamiliar with Scrum, you should read Chapter 1, "Backdrop: The Science of Scrum," for a recap of Scrum theory, flow, practices, artifacts, roles, and meetings. Appendix C, "Resources," provides a list of resources that you might want to access to get a deeper understanding of Scrum.

Appendix D, "Fixed-Price, Fixed-Date Contracts," and Appendix E, "Capability Maturity Model," are the odd ducks of this book. They contain material that might help you use Scrum in rather unique circumstances that aren't described in the case studies that constitute the body of this book.