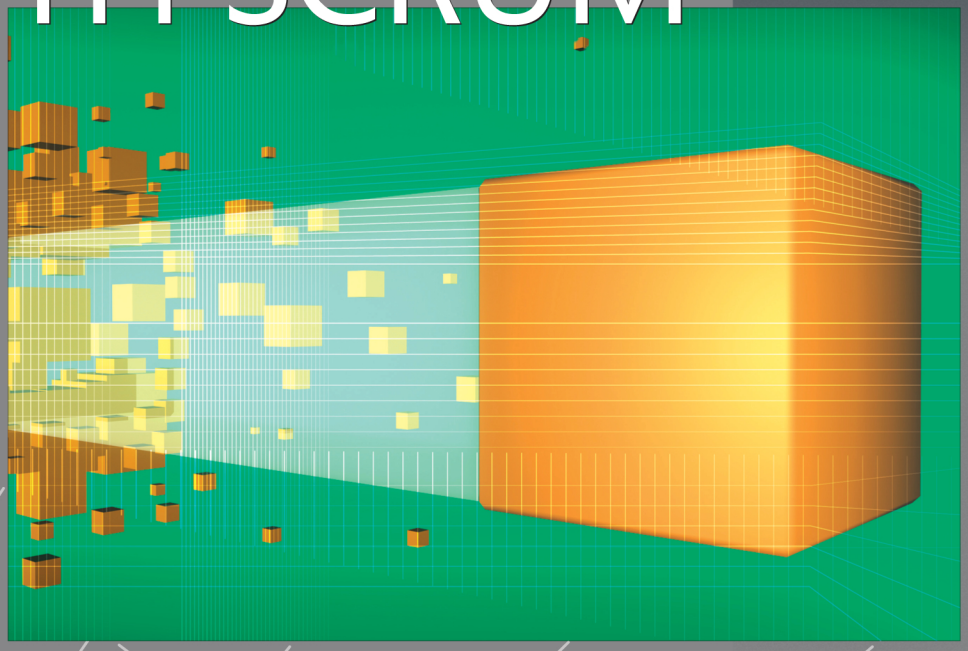
Microsoft

BEST PRACTICES

AGILE PROJECT MANAGEMENT WITH SCRUM



Ken Schwaber

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

ISBN: 978-0-7356-1993-7

Printed and bound in the United States of America.

19 20 21 22 23 24 25 26 27 LSI 8 7 6 5 4 3

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 [2013-03-08]

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.

3

The ScrumMaster

Why did I choose a strange name like "ScrumMaster" for the person who facilitates Scrum projects? Why didn't I continue to use the standard title "project manager"? I wanted to highlight the extent to which the responsibilities of the ScrumMaster are different from those of a traditional project manager. This difference in terminology is symbolic of a drastic change managers must make to their approach if they are to effectively manage Scrum projects.

The authority of the ScrumMaster is largely indirect; it springs mainly from the ScrumMaster's knowledge of Scrum rules and practices and his or her work to ensure that they are followed. The ScrumMaster is responsible for the success of the project, and he or she helps increase the probability of success by helping the Product Owner select the most valuable Product Backlog and by helping the Team turn that backlog into functionality. The ScrumMaster earns no awards or medals because the ScrumMaster is only a facilitator.

Learning basic ScrumMaster practices is easy for most, but some people have difficulty learning the art of being a ScrumMaster. I've encountered some misguided Scrum implementations that don't have as much of an impact as they might have had because the ScrumMaster doesn't understand the philosophy underlying the Scrum methodology. Some ScrumMasters just don't get it, no matter how much they've read about Scrum. Scrum is a simple, straightforward set of practices, rules, and roles, as introduced in Chapter 1 and further described in the Appendixes of this book. But the philosophy behind Scrum is somewhat less simple and can sometimes be difficult to understand. Learning Scrum is a little like learning to ride a bike: after a little bit of time, you just get it—and your muscles get it—and from then on, it's as easy as pie. But until then, you'd better not go riding on major roads. ScrumMasters who don't fully understand Scrum are like novice bicyclists riding down major highways.

As Scrum spreads, I've become more concerned about ensuring that there is an adequate supply of qualified ScrumMasters. I recently received a call from a manager of a production team developing semiconductors for a large company in Texas. He wanted to know about "this Scrum stuff." I asked him what had piqued his interest, and he responded that four months earlier the manager of the design team in Germany had called and said to him, "We've adopted Scrum to manage our design process, so don't expect the usual reports." Yesterday, the same individual had called to tell the Texas manager that the design team had slipped and was three weeks behind schedule. The Texas manager wanted to know, "Is this Scrum?"

This kind of call is all too familiar to me. In another instance, a manager from Brazil came up to me after a class at a recent conference. He was quite excited about the idea of Daily Scrums. He told me he had been using Scrum for more than six months, and he thought implementing a Daily Scrum would really help communications within the team. I couldn't believe that he had read about Scrum but not understood how critical the Daily Scrum is for socialization and synchronization.

These examples show how easy it is for people to misunderstand Scrum. People tend to interpret Scrum within the context of their current project management methodologies. They apply Scrum rules and practices without fully understanding the underlying principles of self-organization, emergence, and visibility and the inspection/adaptation cycle. They don't understand that Scrum involves a paradigm shift from control to empowerment, from contracts to collaboration, and from documentation to code.

Let's look at the experiences of ScrumMasters with differing levels of experience with Scrum. These examples should help us understand how important it is to have a well-qualified ScrumMaster herding the team.

The Untrained ScrumMaster at Trey Research

A consultant is sometimes defined as someone who gives advice more than 100 miles from where he or she lives. I know why this is the case. My neighbors know my lawn has patches and crabgrass in it, just as their lawns do. The police in my town know I sometimes speed. The librarians know I sometimes have overdue books, and they know I have a taste for daring mystery stories. In short, the other residents of my town know I am a regular person with both strengths and shortcomings—I'm not at every moment an expert on all questions.

People often hire consultants because they want to get a different perspective on their situations. This new perspective is often perceived as somehow better than the native view of things. This would be enough of a reason for

clients to think twice before hiring a local consultant. So you can imagine how excited I was when a company in the town where my family has lived for the last 23 years called. The CIO had implemented Scrum and wanted me to check it out.

The company in question was Trey Research, a start-up company that acquires tissue cultures from healthcare organizations and resells them to pharmaceutical companies. Trey Research adds value to the cultures by inventorying and identifying the demographics, illness, and stage of illness represented by each sample. Overloaded with new systems to build and implement, the Trey Research CIO had implemented Scrum. He wanted me to evaluate how Scrum was working at his company and suggest how his implementation might be improved.

What Was Wrong

At the beginning of my visit, I met with the CIO's management team and provided them with an overview of Scrum. We then discussed the various projects under way at Trey Research and how they were using Scrum. Each team had sprinted several times, and everyone was pleased with the changes that had been effected and the progress that had been made.

The ScrumMaster who had been using Scrum the most invited me to attend "his Daily Scrum." The moment I heard this, an alarm bell went off in my head. Why was it "his Daily Scrum" and not "the team's Daily Scrum"? I decided to hold my tongue and wait to find out. He led me to a large room in the basement of the old mansion that was Trey Research headquarters. Nine developers were working at their workstations—five clustered in the center of the room and a pair at each end of the room. From a structural point of view, this was good news: an open work area like this enables the high-bandwidth communication essential for successful teamwork.

At this meeting, the ScrumMaster kicked things off by pulling out a list. Reading from the list, he proceeded to go around the room, asking each person present whether he or she had completed the tasks he had written by that person's name. He asked questions like, "Mary, did you finish designing the screen I gave you yesterday? Are you ready to start on the dialog boxes in it today?" Once he had exhausted his list and spoken to everyone in the room, he asked whether the team needed any help from him. The team members were all silent.

I wasn't sure how to tell him what I thought of his methods. On one hand, work in my hometown was certainly convenient. But how could he have so completely misunderstood all that I had written about Scrum? How had I failed to convey the spirit of Scrum? He turned to me and somewhat proudly asked

what I thought. I paused and then complimented him on the open arrangement of the team room and the general spirit of the team. I then asked him how he knew what the team was working on. He started to say he knew because they were working on what he had told them to work on, but before the entire sentence got out, a look of shock passed over his face. In just a moment of reflection, he had identified the key element of Scrum that he had forgotten to implement.

Lessons Learned

The project manager had read the book on Scrum and learned the mechanics of the Daily Scrum. He had read that team members are supposed to answer three questions at each Daily Scrum:

- What have I done since the last Daily Scrum?
- What am I going to do between now and the next Daily Scrum?
- What is preventing me from doing my work?

However, he was a longtime practitioner of traditional project management techniques. He'd spent years planning tasks and ensuring that teams completed them. Consequently, he had interpreted what he'd read as

- He would check on whether the team members had done what he told them to do since the last Daily Scrum.
- He would tell each member what they should do between now and the next Daily Scrum.
- He would check to see whether he could do anything to help the team accomplish its goals.

To save time, he had shortened the last question into a general inquiry.

The shift from project manager to ScrumMaster had eluded him. He believed that Scrum was merely a series of practices and techniques for implementing iterative, incremental development. He missed the subtle but critical shift from controlling to facilitating, from bossing to coaching. Just as he missed out on these changes, he also missed out on the importance of a self-organizing team. He and the team had committed to a Sprint goal, but the team never self-organized or truly committed to the Scrum goal. The productivity that emerges when a team figures out the best way to accomplish its goals hadn't been realized. Neither did team members have the deep personal commitment that emerges when people puzzle their way through their work on their own. The team's ability to tackle its problems and solve them is the heart of Scrum and

the basis of the Scrum team's extraordinary productivity. Once I pointed this out to the project manager, he immediately saw the error of his ways. "Oh, of course!" he exclaimed. Some people are so embedded in their familiar ways that they have trouble seeing what they have to change, no matter how many articles and books they read and agree with.

The Untrained ScrumMaster at Litware

Litware is a medium-size vendor of planning software. A project management office consisting of one manager, John Chen, and three project managers planned all of the company's releases. After each release was planned, the work was broken down into tasks and organized on PERT charts. These tasks were then divided among the various analysts, designers, programmers, testers, and documenters. The approach was very "waterfall" and very defined. As the complexity of the releases increased and the customer base grew, the release planning phase grew and grew until it was simply unacceptable. The results of each release planning phase were also unsatisfactory: the plans were difficult to adapt to the complexities that the team encountered and to the changes that sales and customers requested.

The company's frustrated managers asked John to work with me to switch the release management over to Scrum. After assessing the situation, John and I set a Scrum start date several weeks out. At this point, we would convert the plans to Product Backlog, provide Scrum training, and then conduct several Sprint planning meetings.

What Was Wrong

During those several weeks, I held a Certified ScrumMaster class and invited John to attend. This was his chance to learn Scrum before he implemented it at Litware. The class prepares people who will be ScrumMasters for projects. As usual, the class was well attended. Unfortunately, there was one conspicuous no-show: John. I kept checking during the day, but he was definitely not there. Later that day, I e-mailed John to find out what had happened. John responded that other priorities at work had precluded his attendance but that we would nonetheless start the Scrum implementation as we'd planned.

I showed up on the appointed day, and we spent the morning laying out the Product Backlog for two teams. In the afternoon, the Litware managers asked me to give an overview of Scrum to the entire development organization. The managers wanted everyone to understand Scrum and what was planned for the two teams. I introduced Scrum and entertained many questions. Everyone

wanted to know where Scrum had been used before, how it worked, and what everyone's new roles would be. They were particularly intrigued by the concept of self-organization because they weren't big fans of task-driven work assigned to them by a project manager. I spent quite a bit of time discussing the shift from project manager to ScrumMaster. I compared the ScrumMaster to a sheepdog who would do anything to protect its flock, or team. We discussed how the team's welfare was the ScrumMaster's highest responsibility and how the ScrumMaster would do anything in his or her power to help the team be productive. At the end of the training session, John and I confirmed the start time with the teams that we were beginning to work with the next day.

I was setting up for the Sprint planning meeting the next morning when Elsa Leavitt, a member of John's staff, arrived to let me know that John had called her and said he would be at an offsite meeting instead of at the Sprint planning meeting. He had sent Elsa along in his stead. John hadn't gotten it: a sheepdog never gets distracted from the flock. John didn't understand that the team would be relying on him. Worse, he had sent a message that Scrum and the team were unimportant to him. He had indicated that he valued offsite meetings more than building software—even though it was the software that was critical to the success of Litware.

I filled in the vice president of development on the situation. He understood the significance of John's absence. He immediately promoted Elsa and appointed her to be the team's ScrumMaster. When the team members arrived for the Sprint planning session, they found that Elsa was their ScrumMaster. She took care of them just as a good sheepdog would.

Lessons Learned

John didn't understand that ScrumMasters have to make a personal commitment to their teams. A ScrumMaster would no more delegate his responsibilities than a sheepdog would lie down for a nap while herding the flock. The team needs to sense that someone is deeply invested in its work and will protect and help it no matter what. The ScrumMaster's attitude should reflect the importance of the project; instead, John's attitude told the team that things at Litware were still business as usual.

I believe that John didn't want to understand the role of ScrumMaster. The behavior of the ScrumMaster is dramatically different from that of people staffing a formal project management office that assigns work and controls its completion. The shift from having authority to being a facilitator was too much for John. Not only is the ScrumMaster role one without authority, but it also potentially represented a career change that John didn't want to make. The ScrumMaster is a leader, not a manager. The ScrumMaster earns the team's respect because he or

she fulfills the duties of the role and not simply because he or she was assigned the role in the first place.

The shift from delegating to being personally responsible is difficult for some people to accept. The "hands-on" aspect of Scrum scares some people. John deselected himself from Scrum by failing to show up for the job. The vice president of development made the right move by reassigning the role of ScrumMaster to someone who recognized its importance.

Overzealous at Contoso.com

Contoso is a software vendor that provides administrative, clinical, and radiology software to healthcare providers. In the late 1990s, a number of dot-com companies were funded to initiate Web-based alternatives to Contoso's products. These new competitors intended to encroach on Contoso by initially offering patient-to-physician portals. These portals would facilitate patient and physician healthcare interactions, including prescription services, healthcare queries, appointments, and online medical advisories. Contoso viewed the portals as Trojan horses through which these competitors would later start offering administrative and billing services as Application Service Providers to Contoso's customers.

To counter this threat, Contoso formed a dot-com subsidiary, Contoso.com. This subsidiary would offer its own patient-to-physician portal, with the difference that its portal would be linked to existing Contoso systems. Several projects were quickly initiated, including development projects, marketing projects, and a public relations project. I was the ScrumMaster for several of these projects, including the public relations project. The public relations project's goal was to increase the marketplace's awareness of Contoso's new strategy and to get current and potential customers to see Contoso.com as an alternative to the other new dot-coms.

Being Right Isn't Everything

The public relations project was very aggressive. In its first Sprint, a public relations firm was hired and a public relations plan conceived and approved. In its second Sprint, Contoso.com and the public relations firm began executing the plan, a key element of which was to make various analyst firms aware that Contoso.com was alive in the Internet space and was a purveyor of Web services. Several analysts had issued reports on this space and not mentioned Contoso in any of them. Many of Contoso's customers were interested in these services but weren't aware that their own vendor was a potential provider.

After considerable effort, the public relations firm was able to set up an all-day session with Contoso.com management and some key analysts. We were to present our plan, our offerings, and our timetable. Our hope was that by the end of the day, these analysts thought of Contoso when they thought of Internet healthcare and healthcare portals.

At the Daily Scrum the day prior to the analyst meeting, one of the team members reported an impediment. I could tell it was going to be a big one from the looks on the faces of all the team members. The vice president in charge of Contoso.com had called for a mandatory offsite meeting the next day. All hands were to be on deck, and all prior commitments were to be canceled. I was incredulous. What could be more important than our Sprint goal, to get Contoso.com visible as a viable alternative to the other dot-coms? The team told me what was more important: the vice president was concerned about morale at Contoso.com and was holding a picnic to improve everyone's mood.

I knew that this was a mistake. The offsite was an impediment to the Sprint. Ironically, it was more likely to hurt team morale than help it. I was certain that the vice president was unaware of the analyst meeting. Otherwise, why would she have insisted on everyone's attendance? To my everlasting amazement, it turned out that she was well aware of the analyst meeting. She even went so far as to ask me to call the analysts and cancel it. She required complete participation at the offsite out of concern that allowing anyone to be absent would encourage everyone to skip out. Unfortunately, I got pretty heated as I was expressing my opinion of this policy. She refused to let the analyst meeting proceed and showed me out of her office.

I was seeing red. I was the sheepdog, and a wolf had attacked the flock. I quickly escalated this impediment to the senior managers. I was sure that they would see the fallacy of the decision and advise the vice president to reconsider. I hadn't anticipated that they would view teamwork as more important than progress and that they would see the sheepdog as an impediment. I was let go shortly thereafter.

Lessons Learned

The ScrumMaster's job is to protect the team from impediments during the Sprint. However, the ScrumMaster has to operate within the culture of the organization. My mistake lay in failing to recognize the value of teamwork to this organization. I had been a consultant for so long that I'd forgotten how much some large organizations cared about not rocking the boat and keeping the corporate family together.

The ScrumMaster walks a fine line between the organization's need to make changes as quickly as possible and its limited tolerance for change.

Whenever possible, the ScrumMaster makes a case and pushes the necessary changes through. The results are often greater productivity and greater return on investment (ROI). However, sometimes these changes are culturally unacceptable and the ScrumMaster must acquiesce. Remember that Scrum is the art of the possible. A dead sheepdog is a useless sheepdog.

Wolves at MegaFund

MegaFund is one of the largest fund management companies in the world. Its innovative funds attracted investors more than the funds at any other organization. However, by 1997, Charles Schwab, eTrade, and other financial companies had revolutionized stock trading. Customers could now manage their own fund accounts, buy and sell stocks, and play the margins without personal assistance from professional stock brokers. The Internet and mobile technology had enabled Web, PDA, cell-phone, and voice-response unit functionality. Unfortunately, MegaFund had fallen behind this revolution. Its technology organization was large, bureaucratic, and cumbersome. To make matters worse, it had implemented Capability Maturity Model Level 3 practices over the last year. If incorrectly implemented, these practices can increase bureaucracy, as they had at MegaFund. MegaFund was now so bureaucratic that it was hard to get anything done.

MegaFund explored ways to enable new technologies that could access the legacy databases where all customer account and trade information was stored. After several false starts, MegaFund managers decided to do it the right way. Usually when managers say that they're going to do a project "the right way," that project ends up dying from excess overhead. Sure enough, after nine months the project was stalled while battles raged over what sort of technology to use. Should it be Solaris, Microsoft Windows NT 4.0, or AIX? Should MegaFund standardize on Intel technology? Were Sun servers more scalable than IBM servers? Was COM the way of the future, or was CORBA the way to go? While these wars were being waged, the competition surged ahead.

MegaFund finally decided to bring in Scrum to break the logjam and get the project moving. Terry Adams, who had been the project manager, had a strong technical background and an intuitive understanding of his new role as ScrumMaster. During the Daily Scrum, he listened carefully to each team member's report. When someone had a problem with his or her equipment, Terry lent a hand. When people were stuck, Terry helped them access expertise external to the project. When purchase orders didn't go through, Terry helped expedite them. He was able to remove impediments without ruffling feathers and without endangering his job.

The Wolves Strike

The team started a Sprint, and within two weeks it had made an impressive amount of progress. The team had selected and begun to use its tools and was implementing the first transactions. By the end of the Sprint, the team would demonstrate an approach to solving MegaFund's technology problems and implementing a suite of competitive solutions.

Russell Hunter, a senior vice president in MegaFund's systems company, was at a cocktail party about this time. After months of trouble, Russ was finally able to brag about some progress. Russ boasted to the head of the electronic funds retail unit, who commented that he had some significant competitive problems that he would like to see solved by this team. Russ, spotting an opportunity to garner some good will, offered to demonstrate a key electronic funds retail transaction at the Sprint review. The next morning, Russ got to the office early and approached one of the systems engineers on the team. The engineer didn't report to Russ. He reported to someone who reported to someone who reported to Russ. Russ was a legend to him, someone who could influence his career with as little as a sidelong glance. When Russ asked him to look into implementing this transaction as part of the Sprint, the engineer couldn't say no.

Something strange happened during that day's Daily Scrum. Terry was listening carefully as usual, so he immediately noticed that this particular engineer reported progress in work that wasn't part of the Sprint goal or selected Product Backlog. Terry asked the engineer to meet with him after the Daily Scrum, at which point the engineer confessed that he'd been asked to do a favor. The engineer was accustomed to senior managers telling him to do something on the side. But Terry knew that this practice was a violation of a fundamental Scrum rule: the team is left alone during the Sprint to accomplish the goals to which it initially committed.

Terry was an intuitive ScrumMaster. He went to Russ and asked about the work that Russ had asked the engineer to do for him. Russ was immediately defensive, knowing that he had violated one of the rules of Scrum. Russ said that it was as though he'd seen a $20 bill on the ground and he couldn't help but pick it up. Instead of criticizing Russ, Terry struck a sympathetic posture. He made it clear to Russ that he understood the importance of this opportunity. However, he said, since Scrum was new to MegaFund, he was sure that Russ was unaware that Scrum had mechanisms for dealing with opportunities like this one. In a case like this, whenever an opportunity arose that was more important than the work selected by the team for the Sprint, management could abnormally terminate the Sprint. The Team, the Product Owner, and management would then conduct a new Sprint planning meeting. The new opportunity would be selected if it truly was the top-priority Product Backlog.

Russ thought about it for several seconds and realized that he didn't want to cancel the Sprint. Everyone would know that he was responsible for halting progress on the project for this minor opportunity. The Sprint planning meeting would make his act highly visible and provide his peers with an opportunity to ask why his pet project was more important than their needs. Russ thanked Terry but demurred, saying that he would meet with the Product Owner and get on the Product Backlog in the next Sprint planning session. Of course, he never did so.

Lessons Learned

Terry used the Scrum rules and practices to keep the project on track. Scrum offers many opportunities to make changes and to respond to new opportunities. Scrum also keeps everything highly visible. The Product Backlog and its prioritization are open to everyone so that they can discuss them and come to the best way to optimize ROI. The Daily Scrum keeps all team activities visible so that the ScrumMaster can enforce the rules and help the team stay on track. By keeping everything in full view, the type of backroom politicking and influence swapping normal in most organizations is minimized. These mechanisms are useful in bureaucratic organizations as a way to get particular things done. But when Scrum is already getting things done, these behind-the-scenes pressures are counterproductive.

Conclusions

At Trey Research and Litware, we saw that it's not always easy to understand the role of the ScrumMaster. At Contoso.com, we saw how a ScrumMaster can self-destruct. At MegaFund, we saw a ScrumMaster both fulfill his responsibilities and embed Scrum practices and rules in the organization. Something unique happened in each situation. The ScrumMaster was aware of Scrum's practices and rules and responded. Sometimes the response was good for the organization, and sometimes it wasn't good. In each instance, the ScrumMaster interpreted the job differently, and the results varied dramatically.

Over the last several years, I've wrestled with the question of how to make the difference between project manager and ScrumMaster, between coach and boss, more readily understood. How can I explain the shift in a way that is easy to absorb regardless of a person's background and inclination? When experienced Scrum practitioners are around to mentor a new ScrumMaster, the transition to Scrum is usually smooth. When I mentor new ScrumMasters, for

example, I can help them understand many of the consequences of failure in part because I've failed so many times! I can also show them the difference between failure and success. We first fill the role of ScrumMaster ourselves, setting an example. Then we invite the new ScrumMaster to begin. We coach the new ScrumMaster after every meeting and throughout the day. We point out opportunities for the ScrumMaster to help the team. We point out ways that the ScrumMaster can tell when the team needs help. We also point out instances in which the ScrumMaster is controlling rather than guiding and explain what the consequences of such acts are likely to be.

The ScrumMaster is responsible for making sure that all the pieces of the Scrum process come together and work as a whole. The Product Owner must do his or her job. The Team must do its job. The chickens must be kept in line. The Product Owner and the Team must collaborate appropriately and use the Scrum meetings for inspection and adaptation.

The responsibilities of the ScrumMasters can be summarized as follows:

- Remove the barriers between development and the Product Owner so that the Product Owner directly drives development.
- Teach the Product Owner how to maximize ROI and meet his or her objectives through Scrum.
- Improve the lives of the development team by facilitating creativity and empowerment.
- Improve the productivity of the development team in any way possible.
- Improve the engineering practices and tools so that each increment of functionality is potentially shippable.
- Keep information about the team's progress up-to-date and visible to all parties.

When the ScrumMaster fulfills these responsibilities, the project usually stays on track. These responsibilities should be enough to keep the ScrumMaster busy; no ScrumMaster should have any time left over to act like a typical boss. Indeed, a ScrumMaster who acts like a program manager probably isn't fulfilling all of his or her duties as a ScrumMaster.

In my experience, some people intuitively understand the ScrumMaster role and take to it like a duck to water. Others struggle to understand Scrum and sometimes make harmful mistakes as they learn. However, even the successful ScrumMaster requires several Sprints to get going. When I am unclear about how to help a Scrum project, I've found it useful to keep the homily "the art of the possible" in mind. Focus on what can be done rather than be frustrated by what can't be done. This thought helps guide my actions at work on projects and in everyday life.

Index

D

E

F

G

H

I

K-L

M

N-O

P

Q

R

S

T

U-V

W

X-Y