Dedicated to ScrumMasters

I thank those who reviewed the book while it was in progress and made many helpful suggestions. They are Mike Cohn, Pete Deemer, Bas Vodde, Colin Bird, Kate VanBuren, Alan Buffington, and Clinton Keith.
I especially thank my wife, Christina, who read, commented, and edited late into many nights.
# Contents at a Glance

Introduction .................................................................................................................. xi

## Part I  Adopting Scrum

1. What Do We Have to Do to Adopt Scrum? ......................................................... 3
2. Scrum qua Scrum ................................................................................................. 9
3. The First Year ...................................................................................................... 13
4. Against Muscle Memory—The Friction of Change ........................................... 21
5. Enterprises in Transition ................................................................................... 29

## Part II  Start Using Scrum for Enterprise Work

6. Organizational Practices .................................................................................... 45
7. Engineering Practices ........................................................................................ 59
8. People Practices ................................................................................................ 69
9. The Relationship Between Product Management/Customer and the Development Team ......................................................................................... 85

## Part III  Appendices

A. Scrum 1, 2, 3 ........................................................................................................ 101
B. More About Scrum ............................................................................................. 113
C. Example Scrum Kickoff Meeting Agenda ............................................................ 119
D. Initial Enterprise Transition Product Backlog ..................................................... 123
E. Scrum Musings .................................................................................................... 127
Table of Contents

Introduction ................................................................. xi

Part I  Adopting Scrum ...................................................... 3
  1 What Do We Have to Do to Adopt Scrum? ......................... 4
     Scrum Requires a New Enterprise Culture ....................... 4
     Prove to Yourself That It Is Worth the Effort ................... 5
     Assess the Type of Change That Will Occur .................... 5
     Caveats ..................................................................... 7
  2 Scrum qua Scrum .......................................................... 9
     Scrum Kickoff Meeting ................................................. 11
  3 The First Year .............................................................. 13
     The First Month ......................................................... 13
     The Second Month ..................................................... 15
     Sources of Transition Backlog Impediments .................... 16
     What If? ................................................................... 17
     The Third Month and Beyond ....................................... 18
  4 Against Muscle Memory—The Friction of Change. ............... 21
     Waterfall Thinking ..................................................... 21
     Command and Control ............................................... 23
     Commitment to Defying the Laws of Nature .................... 24
     Hiding Reality .......................................................... 26
     Summary ................................................................... 27

What do you think of this book? We want to hear from you!

Microsoft is interested in hearing your feedback so we can continually improve our books and learning resources for you. To participate in a brief online survey, please visit: www.microsoft.com/learning/booksurvey/
# Table of Contents

## 5 Enterprises in Transition ......................................................... 29
- Contoso ................................................................. 29
  - Situation ......................................................... 30
  - Application of Scrum ........................................... 30
  - Outcome .......................................................... 31
  - Additional Comments ........................................... 31
- Humongous ................................................................. 32
  - Situation ......................................................... 32
  - Application of Scrum, Phase 1 ................................ 33
  - Outcome, Phase 1 ................................................. 33
  - Situation, Phase 2 ............................................... 34
  - Application of Scrum, Phase 2 ................................ 34
  - Outcome, Phase 2 ................................................. 34
  - Additional Comments ........................................... 35
- Woodgrove Bank ........................................................... 35
  - Application of Scrum ............................................. 36
- Litware ................................................................. 37
  - Situation .......................................................... 37
  - Application of Scrum ............................................. 37
  - Outcome ............................................................ 38
  - Additional Comments ........................................... 40

## Part II Start Using Scrum for Enterprise Work

## 6 Organizational Practices ......................................................... 45
- #1: Organizing Enterprise Work .......................................... 46
- #2: Organizing Enterprise Work for a High-Technology Product Company .............................................. 46
- #3: Organizing Enterprise Work in Other Enterprises .......... 51
- #4: Organizing Enterprise Work for New Systems that Automate an Enterprise Operation ............................. 52
- #5: Organizing the Complexity of Multiple Views ................. 54
- #6: Organizing Work to Optimize Software Product Family Architectures ......................................................... 55

## 7 Engineering Practices ............................................................. 59
- #1: Multilayer System Work Organized by Functionality ......... 60
- #2: Integration of Multiple-Layer Systems ......................... 63
- #3: Integrating the Work of Scrum Teams and Teams Not Using Scrum ................................................................. 66
  - Summary ............................................................ 68
8 People Practices ................................................................. 69
    #1: Organizing People to Do Enterprise Work .......................... 70
    #2: Team Creation ........................................................... 73
    #3: Team Work .............................................................. 75
    #4: How People Are Managed .............................................. 76
    #5: Functional Expertise .................................................. 80
    #6: Compensation .......................................................... 81
    #7: Extra Managers ......................................................... 81
    #8: Teams with Distributed Members ...................................... 82
    #9: Scarce Skills Needed by Many Teams ................................. 83

9 The Relationship Between Product Management/Customer
and the Development Team .................................................. 85
    #1: Shortening the Time to Release Through Managing Value .......... 86
        Relative Valuation with Scrum ......................................... 87
    #2: Just Do It .................................................................... 90
    #3: The Infrastructure, or Core .......................................... 90
    #4: Accelerators to Recovery .............................................. 92
    #5: The Mother of All Problems .......................................... 93

Part III Appendices

A Scrum 1, 2, 3 ................................................................. 101
    The Science ................................................................... 101
    Empirical Process Control ................................................... 102
    Complex Software Development .......................................... 103
    Scrum: Skeleton and Heart .................................................. 105
    Scrum: Roles ................................................................... 106
    Scrum: Flow ................................................................... 106
    Scrum: Artifacts .................................................................. 109
        Product Backlog ................................................................ 109
        Sprint Backlog ................................................................ 111
        Increment of Potentially Shippable Product Functionality ..... 112

B More About Scrum ........................................................... 113
    Scrum Terminology ............................................................ 113
    Scrum and Agile Books ........................................................ 117
    Scrum Books .................................................................... 117
    Books on Techniques Used in Scrum for Managing Product Development .................................................. 117
Introduction

This book is for those who want to use Scrum throughout their enterprise for product development. Right now, you might have pockets within your enterprise that use Scrum, and they are more effective than elsewhere. You are at least partially convinced that using Scrum throughout the enterprise might be a way to make the whole enterprise more effective, but you could use some help in figuring out how to do so. This book is for you.

There are many reasons why your enterprise can’t develop and deploy products and systems as rapidly, inexpensively, and with the quality that you would like. You and your staff probably can already list many of them. Scrum won’t solve them. Scrum is simply a tool that will relentlessly and ruthlessly expose them. As you try to build product within the Scrum framework, every time one of these impediments is reached, it will be exposed in a somewhat painful way. You can then prioritize it and systematically eliminate it. When the impediments are mostly gone, Scrum is a framework that will enable the product development you desire. And it will continue to be your watchdog against any new impediment or old impediments returning home for a visit.

I’ve gathered quite a few experiences and stories as I’ve worked with enterprises adopting Scrum. In this book, I’ve organized them into guidance in the areas that are most problematic. Sometimes this is descriptive; other times I relate the guidance through stories. It is OK that there is no guidance in the other areas. The enterprise should figure out what is likely to work best for itself and try to use it. To the extent that an approach doesn’t work, change it and change it again so that it works better and continues to work better.

Scrum does not prescribe. Scrum includes general guidelines about how to do development and principles to be applied when these recommendation are insufficient. What does this mean? This means that people have to learn to think differently. We want rules to follow, but life and product development are too complex for a single set of rules to suffice in all circumstances. You have to rely on decentralized decision-making, because there probably isn’t one answer for every team any more than there is for every enterprise.

The first three chapters lay out the plan for adopting Scrum. The next two chapters provide insights into some habits that impede adoption and how some enterprises have coped with them. The remaining chapters provide techniques for solving some of the knottier issues. These will help you, but your enterprise’s adoption will be different from anyone else’s adoption. The only common ingredient is people, for better and worse. When people rise to the occasion and work heroically in teams, nothing is better. When they prefer to lay back, play politics, and undercut each other, nothing is worse. You'll get to see both, because Scrum will relentlessly expose everything as you proceed.
Not every enterprise that tries to adopt Scrum will succeed. At times, you and your people will hate Scrum. However, don't shoot it. It is only the messenger. To the extent that you and your enterprise succeed, though, you will always know where you stand. You will know what you can do and can't do. Sometimes such transparency let's us see things that aren't what we wish to see. However, I find knowledge preferable to uncertainty and ignorance. The goal is for you and everyone in your enterprise to wake up looking forward to coming to work, and for your competitors to wish they had never woken up.
Chapter 4

Against Muscle Memory—The Friction of Change

In this chapter:

Waterfall Thinking ................................................................. 21
Command and Control ............................................................ 23
Commitment to Defying the Laws of Nature .............................. 24
Hiding Reality ................................................................. 26
Summary ........................................................................... 27

If you find yourself saying that your group’s developers have satisfied over 29 percent of their customers with successful projects, they are probably relying on best practices, outstanding skills, cutting-edge quality, and a legacy of habits that form intellectual muscle memory. Muscle memory is a deep habit our muscles develop by working together. When the enterprise uses Scrum, the developer’s muscle memory is inappropriate and damaging.

Expect muscle memory to exert itself. When a project is going well, everyone is happy with Scrum. However, when stress, a problem, or an unexpected failure occurs, everyone tends to throw away Scrum and revert to their muscle memory. Teams don’t want to self-manage. They want to be told what to do. Managers don’t want to let teams self-manage. They want to command the teams in all matters, down to the minutest detail. Teamwork is dumped for individual heroics. Quality is abandoned. Everyone draws on what they think has worked best in the past.

Four major muscle memories hinder Scrum’s potential to effect change. They undercut the effort to build products better. Let’s look at them.

Waterfall Thinking

The waterfall process emerged from project managers’ wishes to overcome complexity with predictability. It has been the predominant development process used over the last 25 years. Waterfall is taught in universities, it’s described in most process books and other literature as the correct approach, and the Project Management Institute has formalized it. Every project manager knows waterfall deep in his or her bones and feels it is correct. Habits accrued from waterfall development are embedded in enterprises. I call this “the tyranny of waterfall”; it is

Part I Adopting Scrum

inescapable. Even people who don’t know it as the waterfall process think of it as the “right” way or “the way we’ve always done things.”

When some people are asked to use Scrum, they are profoundly uncomfortable. It goes against the grain and feels risky. They reply, “Yes, but...” because their trained response is to prefer the waterfall practices. For instance, requirements are handled very differently with Scrum. About 50 percent of a typical project is spent developing requirements, architecture, and design. During the same project, 35 percent of the requirements change and over 65 percent of the functionality described by the requirements is never or rarely used. Regardless, in waterfall, all requirements, architecture, and infrastructure are fully detailed before the team builds functionality.

Scrum views requirements and architectures as inventory. Inventory is a liability because if some requirements change or aren’t used, the time spent to understand them or design for them is a waste. The Product Backlog, which lists the requirements of Scrum, only has to be defined in time for a Sprint Planning Meeting; the work to fully understand it is performed only as the Sprint that transforms it into product occurs. Requirements are developed and architecture emerges in the Sprint for which the Product Owner requests them. To someone steeped in waterfall thinking, this practice is imprudent, risky, and reckless. To develop code from incomplete requirements, they know, is just asking for trouble. A waterfall architect told a Scrum architect that the only way to build a solid architecture was to think it through up front, before any code was built. The second architect said he thought that building it as requirements emerged might create a more stable architecture because it would be proven, piece by piece.

Let’s look at the implications of another waterfall habit, functional specialization. The Product Owner discusses the Product Backlog with the Scrum team. Together, the team members discuss the requirements and create designs, code, tests, and documentation. A waterfall traditionalist believes, however, that only a designer can design, only a programmer can code, only a quality assurance (QA) person can test, and only a technical writer can write documentation!

I was attending a Sprint Review Meeting. The Scrum Team had selected five items from the Product Backlog for the Sprint. Only one item was finished. The team members said that the QA (testing) people on the team hadn’t completed their testing. However, a Scrum team is cross-functional. The entire team is responsible for building completed pieces of functionality every Sprint. It wasn’t the QA people who didn’t finish the testing—the Scrum team didn’t finish the testing. Scrum counts on everyone chipping in their best effort to do the work. When functional expertise is necessary, the people with those skills take the lead, but anyone can do the work.

Trey Research (TR), our first hypothetical company, develops acoustic products. TR was ready to introduce a new radio. Thousands were in the warehouse ready for shipment. Dr. Trey is the founder and CEO. As Dr. Trey read the user manual in his office, his frown got deeper and
deeper. Finally, he called the technical writing manager, Matthias Berndt. Dr. Trey said he was very disappointed in the documentation; it was unusable. Berndt agreed, but said that it accurately reflected the way the radio worked. Dr. Trey kept his calm as he asked Berndt to go to the warehouse, open a radio box, and see if it worked the way the user documentation indicated. Two hours later, Berndt appeared in Dr. Trey’s office with an open box and the user manual. Berndt said, “Much as I hate to say this, Dr. Trey, the manual accurately reflects the radio’s operation.”

Dr. Trey lost his temper. He asked Berndt how he could have let such a terrible radio be built. Didn’t Berndt know that the radio was unacceptable? Berndt agreed, but he said that he had nothing to do with the radio until after it was built. Dr. Trey grew even more troubled and asked, “You mean, even though you’ve worked here 23 years and know our radios inside out, you don’t have anything to do with their design? You only document them after they are built?” Berndt confirmed this. This dysfunctional approach was the impetus for TR to adopt Scrum. Now everyone on a cross-functional team at TR designs the radios. Dr. Trey knows that if the radio’s design doesn’t meet the approval of the engineers, technical writers, and testers, it shouldn’t be built.

**Command and Control**

Workers are best able to figure out how to do their work, not their managers. The work is complex and has unexpected nuances. If workers are bound by someone else’s instructions, they aren’t free to do the work the best way possible.

Attendees at the Certified ScrumMaster class examine the productivity of self-management through an exercise. First, a contained space of approximately 400 square feet is established. Chairs, tables, and other obstacles are liberally sprinkled throughout the space Everyone is placed in a pair, each pair consisting of a boss and a worker. The exercise is for the bosses to get their workers to take 60 full steps in two minutes using the commands of start, stop, left, right, faster, and slower. At the end of two minutes, about 50 percent have gone 60 paces. The rest have gone fewer paces. In the second part of the exercise, pairs are broken up. Everyone is a worker who manages his or her own activities. Each is free to use the previous commands or come up with more appropriate commands. Everyone is asked to take 60 full steps and then stop. Everyone is done within one minute. The self-management of the second exercise has doubled productivity. And because managers are now also workers, productivity has quadrupled.

Certified ScrumMasters know that self-managing Scrum teams are more productive. The front 10 percent of their mind is sold on self-management. But the back 90 percent knows that they are still in charge. If anything goes wrong, they will step in and tell the team what to do. We have been trained that this is the best way to absolutely make sure things go right. The command and control habit is very difficult to discard.
Part I  Adopting Scrum

It takes time for Scrum teams to gel and start performing. Some teams require more support than others. The ScrumMaster is responsible for teaching self-managing teamwork to the team. For instance, if the team comes to the ScrumMaster saying, “This Product Backlog item is too large for one Sprint! What do we do?”, it isn’t told the answer. Instead, the ScrumMaster leads the team through the process of figuring out how to deconstruct the backlog. The ScrumMaster teaches; the team learns and finishes the exercise. The next time a similar situation arises, the team will know how to act independently. The moment the ScrumMaster tells the team what to do and how to do it, he or she exerts command and control. In command and control, the ScrumMaster believes he or she is responsible for productivity and problem solving. In self-management, the manager thinks that he or she is responsible for teaching the team self-management and problem solving.

One project that I initiated included more than 50 developers. New development had to be done in conjunction with maintenance of the existing system. A reasonably good Product Backlog was in place. I spent several days reviewing employee files and resumes, as well as talking with the managers, trying to decide the best team composition. After those several days, I had a headache. So I called all the developers into the room. The Product Owner reviewed with the developers the upcoming project and the Product Backlog. I described the rules for composing Scrum teams and determining their size. We then asked the developers to organize themselves into teams. We told them the teams didn’t have to be permanent but they should give it their best shot. Within four hours, they had formed their own teams. The teams created agreements among themselves about how the teams would cooperate. During the next Sprint, several team members shifted to other teams. At the end of that Sprint, the developers told us they were pretty happy with the team composition. They asked if they could continue to change as needed, however. We, of course, gave permission—we didn’t have any better ideas!

Commitment to Defying the Laws of Nature

I live in Boston and frequently work in New York City. In just 45 minutes, the Delta Airlines shuttle can take me from Boston’s Logan Airport to New York’s LaGuardia Airport. I sometimes pack more than one meeting into a day because of this convenience.

One day, I was up first thing in the morning and down to New York for a meeting. I got back to LaGuardia by 2:00 P.M. to catch the 2:30 shuttle to Boston. I had an end-of-day meeting in downtown Boston at 4:30 P.M. This schedule would have worked, except LaGuardia was fogged in and all the afternoon flights were delayed or canceled. I went over to the Hertz counter. I told the Hertz clerk that I needed to be in Boston in 90 minutes and wanted a car. She looked at me strangely. Apparently, my need couldn’t be met. The laws of the road, the top speed of the cars available, and the distance between Boston and New York City made my requirement pitiable and impossible to satisfy. The laws of physics thwarted my wishes.
Chapter 4  Against Muscle Memory—The Friction of Change

Now consider a Product Owner at TailSpin (our next hypothetical company) who has met with her Scrum team prior to the first Sprint. She handed out a presentation with 12 bullet items. She told the team the 12 items had to be done and the release needed to ship within six months. The team looked blankly at the Product Owner and told her that, even without knowing more details about the project, it was impossible to do. The Product Owner answered, “If we don’t deliver these features by then, we cannot sell the product, so it has to be done.” Just like me in New York, this Product Owner needed something that wasn’t possible.

Business runs on commitments. When you make a commitment to someone else, you have given your word. The other person arranges his business accordingly, counting on you to do what you say. This understanding is based on trust and is a tremendous source of efficiency. Let’s give ourselves a short test on commitment. Read the following exercises and see if you can commit to fulfilling the other person’s needs.

■ Someone asks you to commit to having some item built for them. She asks you for the date on which it will be finished and for the price that it will cost. You spend some time with her trying to understand exactly what she wants, but the details are elusive. Also, you are going to have to handcraft this thing. You aren’t sure of the exact skills of your workers or their availability. Also, the flu has been sweeping the town, and it could hit your team. The technology for building this item has worked so far, but a new release is coming out with mixed reviews. The person asking for the commitment also tells you that she might need to change some things along the way. Do you commit?

■ Someone tells you that he wants a product by a specific date. You must do this thing because he has already committed the product by this date to somebody else. He wants you now to back up his commitment with your commitment. You aren’t sure exactly what the whole commitment is, but the other person has power over your career and salary. Do you commit?

Of course, it is impossible to openly commit in either circumstance. You just don’t know. You might feel that you have no choice but to commit in the second instance, but you had better have some tricks up your sleeve in case you get in trouble.

Pressuring someone to commit to an outcome regardless of what he or she believes is possible is a bad habit. If the person under pressure is honest, she won’t promise anything. If she is cornered, she might make an undeliverable commitment. Neither alternative—a lack of commitment or a false commitment—is helpful if you need something to happen. Our muscle memory tells us that we can ask our engineering team for a commitment. The engineering team’s muscle memory is to provide one, regardless of the circumstances. Where the waterfall process is in vogue, we have no choice but to do so. But we have other options when Scrum and iterative, incremental processes are used. These Scrum alternatives are presented in depth in Chapter 9, “The Relationship Between Product Management/Customer and the Development Team.”
Hiding Reality

Our next hypothetical company is Coho, one of the largest resellers of cars in Europe. Senior management was rolling out Scrum to improve its ability to introduce new capabilities to customers. In the first Sprint of the first project, the Scrum teams delivered more functionality than they had committed to. Everyone, from senior management to the customers, was excited and pleased.

For the second Sprint, the Scrum teams committed to a large amount of Product Backlog. Two weeks into the Sprint, the teams realized they were in trouble. When the teams got together, they all had the same story: the functionality was significantly more complex and difficult than the first Sprint. Of the 24 pieces of functionality the teams had committed to, they figured that they might complete 7 or 8. After the way everyone had cheered them on at the first Sprint Review, they feared what would happen if only 33 percent of their second Sprint were done. The teams decided the only way they could deliver everything was to drop testing and refactoring; they would just slap the new functionality on top of the old. They figured that by committing to far less for the third Sprint they would have time to go back and fix it all.

One of their ScrumMasters asked them what they were doing. The ScrumMaster realized that Scrum is about empirical progress and transparency, so the Product Owner always knows what is going on and can make the best decisions. Wasn’t the approach the team decided to take hiding things from the Product Owner? Weren’t they pretending that things were done when they weren’t? The teams, after expressing their fears that the Product Owner might fire all of them, went to the Product Owner and showed him where they were and what problems they were running into. The Product Owner looked at them and said, “I knew you overcommitted. I was going to ask you what was going on. I hoped maybe you knew something that I didn’t. Well, I’m really glad you came to me.” The Product Owner and teams reduced the commitments to match their new findings and proceeded, Sprint by Sprint, to build a great new system.

When I discuss this kind of fear at courses I teach, the attendees’ own fear is palpable. The soon-to-be Scrum users don’t think that transparency, or truth, is acceptable where they work. They tell me that they will be fired if they tell the truth. Truth isn’t what their customers want to hear. They tell me their customers will find someone else who will lie to them if they don’t. I have seen this in class after class for five years. People in product development think that their customers want to hear news only if it is good news and would rather hear a lie than the truth. “Lying” is a harsh word. But what else do you call saying that something is true when you know it not to be true? What else do you call misleading someone with information or holding back information that would have led them to better decisions? The Product Owners want to believe in magic, and the developers support the belief by lying. “Can you do this project by this date?” “Sure, no problem.”

The developers are aware of the complexities that cause changes to their original estimates. They are aware that the customer is unhappy. If a project manager is approached by a
customer 60 percent of the way through a project and asked how the project is going, the project manager doesn’t really know. She knows that some things are going well. She also knows that some things are not going so well. She also knows that she hasn’t checked up on some things that could prove critical. However, saying “I don’t know” is unacceptable, so project managers have learned to say, “Right on,” “Right on target,” “Piece of cake,” or anything equivalent that will get the customer to go away and leave them to try to get everything on time, on cost. Basically, they lie. It is simpler than exposing all the nuances and complexities that add up to “I don’t know.”

Project managers might also believe that lying saves time. But because Scrum relies on transparency, misrepresentation undercuts the entire application of Scrum. If the Product Owners do not know exactly where things stand at any point in time, they will be unable to make the best decisions possible about how to achieve their goals. They need the best information possible, whether they view it as good or bad.

**Summary**

The iterative, incremental nature of Scrum causes change within the enterprise. The enterprise must adapt to monthly project changes, not just change at the very end. A project produces potentially usable increments of the whole system every month. Teams produce complete pieces of that increment daily. This frequency of completed work causes change.

Dysfunctional behavior that was hidden becomes visible. Problems caused by the dysfunctional behavior are magnified. As you solve the dysfunctional behavior, don’t think that the solution is complete. For 25 years, every habit described in this chapter has provided better solutions to people in your enterprise than anything else has. Now these people are going to try something better, something that even feels right. But when the problems of product development and management arise, your people are going to feel naked. They haven’t accrued muscle memory in these new ways yet. So, because it feels safe—just for now—they return to these habits, the old-reliable habits. Your enterprise and its people will take four steps forward, three steps back, two steps forward, one step back. They will continually progress, but they will bemoan their inability to ignore and transcend old habits. Scrum, however, won’t them let ignore the consequences of these habits.
When your enterprise uses Scrum, you can monitor all development every Sprint. You can redirect enterprise work to take advantage of new opportunities and maximize enterprise return on investment (ROI). The entire enterprise can change course quickly. To be able to do these things, you must have all your enterprise’s work in a single Product Backlog. Creating such a backlog can take over one year and is very difficult. Once it’s done, however, you’ll wonder how you managed previously. Without an integrated picture of all of the enterprise’s work, it is impossible to assess progress and perform impact analyses.

In this chapter, I’ll explain how to create such an enterprise Product Backlog. An overview is presented in the “#1: Organizing Enterprise Work” section. The enterprise Product Backlog structure is somewhat different for high-technology product enterprises than it is for an enterprise that deploys technology to make its operations more competitive. We’ll look at high-technology Product Backlogs in the “#2: Organizing Enterprise Work for a High-Technology Product Company” section. In “#3: Organizing Enterprise Work in Other Enterprises,” we’ll look at creating a Product Backlog for other enterprises.

Another Product Backlog variant is organizing work when a new enterprise operation, including systems that automate it, is being developed. This scenario is discussed in “#4: Organizing Enterprise Work for New Systems that Automate an Enterprise Operation.”

A Product Backlog is the work of the company. Many views of this work are often required. The “#5: Organizing the Complexity of Multiple Views” section shows how to correlate and manage multiple views. The information in this section will help you handle some complexities of maintaining multiple views.
Finally, we’ll look at how to organize work if your enterprise is using a software product family architecture to optimize reusability in “#6: Organizing Work to Optimize Software Product Family Architectures.”

#1: Organizing Enterprise Work

Scrum seems to organize work into Product Backlogs. But how do I organize my entire enterprise’s work into a Product Backlog and what are the benefits of doing so?

We can organize all of an enterprise’s development work into an enterprise Product Backlog. To create an enterprise Product Backlog, create an enterprise view of all projects and programs. These views are top-down decompositions that organize the Product Backlog by enterprise product architecture, organization, or programs. If the enterprise sells high-technology products, use a product decomposition that consists of the following information: product family, product, features, function, and task. If the enterprise uses technology to automate its products, like a financial institution does, use details of the organizational structure. The rest of this chapter presents ways of creating these views and linking them to each project’s Product Backlog. As we correlate and link the detailed Product Backlog of Scrum projects to the enterprise view, the enterprise Product Backlog starts taking form. We then fill in the enterprise Product Backlog as more projects are started. You must eventually identify, organize, and prioritize all current and planned work.

To the degree that all the work of the enterprise is in an enterprise Product Backlog, you can track the progress of every program, release, and project through burn-down charts. For any area of interest, a burn-down chart tracks progress toward a release goal across time. With burn-down charts, you can assess the impact various projects and programs have on each other and on the enterprise. You probably will be unpleasantly surprised. Programs that you thought were well underway might be behind. You might find that splitting people across many projects has slowed overall work rather than allowing the enterprise to take on more. You will get a lot of information, some confirming your hopes and others dashing them. You will, however, have solid information with which to manage the enterprise.

#2: Organizing Enterprise Work for a High-Technology Product Company

My enterprise builds products that we sell to external customers. Scrum organizes work into Product Backlogs. How do I organize my enterprise’s work? In particular, if I have an opportunity to do something new, how do I quickly reorganize to do so?

A Product Backlog can represent all known development work for an enterprise’s products. The products decompose into features, functions, activities, and tasks, reflecting the product structure and terminology. A Product Backlog defines the changes that are needed at this
lowest level. This decomposition can be aggregated into product families and all of the enterprises’ development work, as shown in Figure 6-1.

<table>
<thead>
<tr>
<th>Product Family</th>
<th>Product</th>
<th>Feature</th>
<th>Function</th>
<th>Activity</th>
<th>Backlog</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personal Finances</td>
<td>......</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corporate Taxes</td>
<td>......</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Personal Taxes</td>
<td>Whirl-Wind Deluxe</td>
<td>Personal Information</td>
<td>About You</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Filing Status</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Personal Information</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Location</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mailing Address/Phone</td>
<td>User must be able to type in different format telephone numbers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>State of Residence</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 6-1**  Enterprise Product Backlog

A product or system architecture consists of modules or components at the lowest level of decomposition. One or more of these components will be changed to satisfy a Product Backlog item. We can organize a separate Product Backlog for product functionality common to more than one product. This Product Backlog’s structure reflects the system’s architecture, as shown in Figure 6.2. Overall prioritization for the good of the enterprise is mandatory. The Product Owner of the common functionality has to be someone with return on investment (ROI) responsibility for all enterprise products.

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Activity</th>
<th>Task</th>
<th>Module</th>
<th>ID</th>
<th>SPF Prty</th>
<th>SPF Size</th>
<th>CI Prty</th>
<th>CI Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen User Interface Controls</td>
<td>Formatted Numeric Entry</td>
<td>Domestic Telephone Number</td>
<td>C413</td>
<td>72</td>
<td>2</td>
<td>61</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Business Logic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Base Controls</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Base</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 6-2**  Common infrastructure Product Backlog of requirements
Let’s look at how we could respond to a customer requiring enhanced functionality in the Corporate Taxes product family. We estimated the effort to make the enhancement at 100 points of work. (A point of work is an arbitrary measure.) The customer needs it within six months. We are in the fifth month of our enterprise’s annual plan.

An enterprise burn-down chart shows the annual baseline plan, as shown in Figure 6-3.

![Annual Road Map](image)

**Figure 6-3**  Burn-down of baseline roadmap plan

We assess progress against the plan. The plan is maintained in an enterprise Product Backlog. The measurement is against the most current plan, which is usually different from the baseline plan. In the fifth month, we can compare the currently planned functionality against that which has already been delivered, as shown in Figure 6-4.

The difference between the two plans represents the degree to which the enterprise is ahead of or behind plan. Figure 6-4 shows that we are behind our plan and behind on our commitments.

![Enterprise Actual vs. Planned Work](image)

**Figure 6-4**  Burn-down of enterprise actual vs. plan
At the end of the fifth month, the plan committed us to have 1214 points of work left. Instead, there are 1320 points of work left to be completed. If we add the new 100 points of work requested in the Corporate Taxes product line, the planned versus actual measurement becomes worse, as shown in Figure 6-5.

**Figure 6-5 Burn-down of actual vs. plan with new work added**

The planned work is the bottom trend line. The actual work left without the additional work taken into account is the middle trend line. The top trend line shows actual work remaining if the new work is committed to. All of these trend lines have been projected to year end to show the probable gap between planned and actual work.

To take on the additional Corporate Taxes enhancements, we need to decommit to other work. We could increase costs through additional new hires, but productivity drops as new people are brought on board and increases only after six or so months. We need to find some other work that we can defer. First, let’s add the new work to the Corporate Taxes part of the Product Backlog. It is the fifth row of Figure 6-6. We then estimate and prioritize it compared to all other work in the enterprise’s Product Backlog. For Scrum estimation techniques, see Mike Cohn’s recent book, *Agile Estimating and Planning* (Prentice Hall, 2004). The prioritized enterprise Product Backlog (summarized) looks like Figure 6-6.

We need to accommodate 206 new points of work (100 new points of work added to the current shortfall of 106 points). We can decommit lower priority work. The first item to put on hold is the lowest priority in the bottom row: Personal Taxes, State of Residence, Item 6. The remaining 148 points of work to be deferred (206 needed less the 58 points of Item 6) has to come from the Personal Finances product, the next lowest priority. Its entire workload has 1048 points of work planned for the year.
When we drill down and look at the burn-down for the Personal Finances product line, it is ahead of plan. We then drill down into its work to see where we can free up some effort while minimizing the impact. In Figure 6-7, we drill down to look at just the work for Personal Finances.
The Personal Finances work is ahead of schedule. At the end of the fifth month, we had planned to have 217 points of work left, but only 160 remain. We are 57 points ahead of plan. We might be able to use this capacity for the new work in the Corporate Taxes product line.

Drilling down in the Personal Finances work, we can see which specific areas are ahead of plan. Then we can assess whether the people doing that work are skilled and capable of helping the Corporate Taxes product. If they are, we might be able to redeploy them. We will ask the Product Owner of the Personal Finances product line whether he or she can form a new team that can be reassigned for four months.

This exercise took care of the new work and enabled us to get the new customer’s business. We assessed the enterprise’s ongoing work to identify excess capacity. We could do the same thing every month to detect shortfalls and slippages.

As priorities change and new opportunities occur, we can realign our work to maximize enterprise ROI. The Product Owners at every level of the enterprise are able to track their work against their commitments. We can shift the enterprise to take advantage of new opportunities while assessing and then tracking the impact.

#3: Organizing Enterprise Work in Other Enterprises

*My company uses our Information Technology organization to develop software for my line operations. This software makes the operations more effective. How does the management of these operations use Scrum, or do I leave this to the Information Technology department?*

Product Owners are the managers of their operations. They define work to enhance their products in the Product Backlog. The development work can be to enhance automated systems or manual operations. Training and implementation work is also part of the Product Backlog. The Product Backlog is sorted by System and Priority to organize work within the Information Technology (IT) organization. IT teams are formed based on Product and System identifiers.

We can use the following example of a banking enterprise to see how to do this. A bank sells financial products to its customers. It is organized into lines of business (LOB). Each line of business consists of operations that sell and service financial products. These operations are automated through internal systems. For instance, a bank can have a Trust LOB, a Commercial Banking LOB, and a Consumer Banking LOB. Within the Consumer Banking LOB is a Teller operation, a Loan Creation operation, and so on. These are serviced by a Product Development and Management department that devises the various financial products. Each operation is supported by one or more computer systems. As new products are conceived, the operations and systems supporting them must be developed or enhanced. The Product
Backlog, or requirements, to do so are organized by LOB, operation, activity, and task. Figure 6-8 represents such a decomposition.

<table>
<thead>
<tr>
<th>Enterprise</th>
<th>Line of Business</th>
<th>Operation</th>
<th>Product</th>
<th>Activity</th>
<th>System</th>
<th>Component</th>
<th>Requirement</th>
<th>Priy</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank Trust</td>
<td><strong>Corporate Banking</strong></td>
<td><strong>Consumer Banking</strong></td>
<td>Teller</td>
<td>Mortgage</td>
<td><strong>Savings</strong></td>
<td>Deposits</td>
<td>Teller31</td>
<td>C524</td>
<td>33</td>
</tr>
<tr>
<td><strong>Customer can make a deposit across accounts</strong></td>
<td><strong>Customer can perform deposit themselves using new automated teller terminal</strong></td>
<td><strong>Withdrawals</strong></td>
<td><strong>Checking</strong></td>
<td><strong>Plat</strong></td>
<td><strong>IRA</strong></td>
<td><strong>Filing Status</strong></td>
<td><strong>401K</strong></td>
<td><strong>Personal Information</strong></td>
<td><strong>Mortgage</strong></td>
</tr>
</tbody>
</table>

Figure 6-8 Financial enterprise Product Backlog

#4: Organizing Enterprise Work for New Systems that Automate an Enterprise Operation

We are building a new system for a division in our enterprise. It will replace a patchwork, older system. How can the work be directed by the Chief Operations Officer of that division so that it makes sense to her, while being organized and prioritized in a way that makes sense from a systems architecture viewpoint?

Data is the business of some enterprises, such as credit reporting, encyclopedias, news, and mapping. These enterprises acquire, format, and sell data. Enterprises sometimes need to build entirely new systems for these type of operations. The managers of these operations need to correlate and prioritize developing a new business operation with building new systems to automate it.
The business operation is organized into several primary functions. The data is acquired. The data is continually groomed to provide additional value through new relationships and attributes. The data is managed for accuracy and quality. The data is extracted for sale to customers. Some extracts are periodic, while others are continuous. At the lowest level of the business operation, activities and tasks are performed. These tasks are manual, manual with automated assist, or completely automated. The automated system is organized as an architecture that has nonfunctional attributes such as performance, scalability, security, and workflow.

The person who runs this operation is the Product Owner. He or she is responsible for overall profitability and the long-term investment in the new system. He or she is responsible for prioritizing the development to support a phased, secure implementation as well as for meeting technical dependencies. As an example of technical dependencies, the workflow framework might be essential to have in place prior to implementing acquisition and editing functionality.

The intersection of operational and systems decomposition is shown in Figure 6-9. The Product Backlog work occurs at the intersection.

<table>
<thead>
<tr>
<th>Divisions</th>
<th>Departments</th>
<th>Sections</th>
<th>Subsection</th>
<th>Activities</th>
<th>Tasks</th>
<th>Product Backlog</th>
<th>Component</th>
<th>Module</th>
<th>Sub-system</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations</td>
<td>Acquisition</td>
<td>TCX01</td>
<td>TC201</td>
<td>TCX01</td>
<td></td>
<td>TCX01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grooming</td>
<td></td>
<td>TCX01</td>
<td>TCX01</td>
<td>TCX01</td>
<td></td>
<td>TCX01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Management</td>
<td></td>
<td>TDX01</td>
<td>TCX01-01</td>
<td>TDX01-02</td>
<td>TDX01-03</td>
<td>TDX01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Management</td>
<td></td>
<td>TDX01-01</td>
<td>TDX01-02</td>
<td>TDX01-03</td>
<td>TDX01-04</td>
<td>TDX01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Management</td>
<td></td>
<td>TDX01-02</td>
<td>TDX01-03</td>
<td>TDX01-04</td>
<td>TDX01-05</td>
<td>TDX01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Management</td>
<td></td>
<td>TDX01-03</td>
<td>TDX01-04</td>
<td>TDX01-05</td>
<td>TDX01-06</td>
<td>TDX01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 6-9** Intersection of operational and system views in a Product Backlog
The Product Backlog item “Display areas to be selected” is part of the operation’s Data Management function. It is used by the supervisor of the Referential Integrity section to frequently inspect and check data referential integrity. The new system has a component, CSetup04 (which is part of Subsystem TDX01-05 and System TDX01), to automate this.

The operational viewpoint also uses Product Backlog items to describe work to enhance a work activity, including creating documentation and retraining. It includes columns that reflect operational implementation priorities and efforts. The systems view includes a column for the effort to build the component and the priority in which it will be developed. The systems view also includes Product Backlog items for systems that provide infrastructure used by the other systems, such as workflow. Other work, such as constructing distributed development environments and upgrading the production environment, have their own Product Backlog items. This Product Backlog is prioritized according to the most logical sequence for developing the system.

#5: Organizing the Complexity of Multiple Views

I’ve seen how to create several views of an enterprise Product Backlog. But there are some complexities you haven’t discussed. Can you describe how to handle them?

Product Backlog is a prioritized list of work. We can relate it to three areas: its occurrence in a product or system, its occurrence in improving a business operation, and its occurrence in systems architecture. We can then create complex views by intersecting these relationships. Figure 6-9, seen in the previous section, shows an example of several views of a Product Backlog. It shows the relationship of a business operational view (Divisions, Departments, Sections, Subsection, Activities, and Tasks columns) to the work in a Product Backlog (Product Backlog column), which is then related to the systems architecture view (System, Subsystem, Module, and Component columns).

Product Backlog items range from small to big. Small items usually relate to fine-grained business operations, system architectural components, or product tasks, as shown in Figure 6-9 earlier. As the items increase in size, the corresponding items they relate to increase in size. For instance, a Product Backlog item referred to as “Automatically flow applications from investigation to acceptance and notification” relates to subsystems, business activities, and product themes. It is large and high level.

Modules or components are often used by more than one operational task or product activity. The Product Backlog item to change a component then has to be entered one time for each
time it automates the task or activity. However, it is estimated for only one of the occurrences. All occurrences inherit the highest priority need and are scheduled accordingly. Sometimes multiple occurrences of a Product Backlog item are indicated in one column in the spreadsheet.

#6: Organizing Work to Optimize Software Product Family Architectures

Some enterprises develop products and families of products. Some of the functionality is product specific, but other parts are shared among all products. How is this work organized with Scrum?

Many enterprises have more than one product. They often separate common functionality into a component infrastructure library to simplify defining new products or enhancing an existing product. When products are developed, some components are unique to the product, but other components might already be in the infrastructure, reducing overall development time and costs. If some potentially common functionality isn’t already in the infrastructure, it is developed there to reduce the costs for future products. By keeping the infrastructure in good shape and well cataloged, new product development is simplified.

The role of the Product Backlog needs to be extended to address this common infrastructure. The Product Backlog usually just lists requirements of work to be done for a product. Now the Product Backlog will reflect the structure of the entire product family. The product family decomposes into products, features, functions, and activities, as shown in Figure 6-10. When something new is needed, the requirement is added. Some Product Backlog requirements will be satisfied by components or databases in the common infrastructure. Figure 6-10 demonstrates this by using the “Common” designator in the Domain column. If this is an existing component that needs enhancing, the ID for the existing component is recorded. When the Product Backlog is sorted by requirement priority and requirement, it starts with a prioritized list of work to be done.

The common infrastructure supports all products. It has its own Product Backlog. This is organized by aspect. This backlog is populated with maintenance work and all work requested for each Product Family and Product, as shown in Figure 6-11.
Part II  Start Using Scrum for Enterprise Work

<table>
<thead>
<tr>
<th>Product</th>
<th>Feature</th>
<th>Function</th>
<th>Activity</th>
<th>Backlog</th>
<th>ID</th>
<th>Domain</th>
<th>Prty</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>WhirlWind</td>
<td>Personal</td>
<td>Information</td>
<td>About You</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Special</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Filing Status</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Personal Information</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Location</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mailing Address/Phone</td>
<td>User must be able to type in different format telephone numbers</td>
<td>C413</td>
<td>Common</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>State of Residence</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Other State Income</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Occupation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Phone Listing Option</td>
<td>User must be able to type in different format telephone numbers</td>
<td>C413</td>
<td>Common</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Create User ID</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Hurricane Katrina</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Special Situations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Dependents</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Import Information</td>
<td>Import from Last Year</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Dependents</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Import Your Information</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Federal Taxes</td>
<td>Income</td>
<td>Wages and Salary</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6-10  Software product family Product Backlog of requirements

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Activity</th>
<th>Task</th>
<th>Module</th>
<th>ID</th>
<th>SPF Prty</th>
<th>SPF Size</th>
<th>CI Prty</th>
<th>CI Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen User Interface</td>
<td>Controls</td>
<td>Formatted Numeric Entry</td>
<td>Domestic Telephone Number</td>
<td>C413</td>
<td>72</td>
<td>2</td>
<td>61</td>
<td>1</td>
</tr>
<tr>
<td>Business Logic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Base Controls</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Base</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6-11  Common infrastructure Product Backlog of requirements
The Product Owner for all product families prioritizes the infrastructure Product Backlog. Only this person can evaluate all product family priorities against each other and against the need to maintain and sustain the common infrastructure. This priority is maintained in the Common Infrastructure (CI) Prty column. The relative size of the work, as evaluated by the infrastructure teams, is maintained in the CI Size column. This work might be different in size than that estimated by the Product Team. Note that the duplicate Product Backlog requirements from Figure 6-11 have been merged into one.
Index

A
accountability, compensation and, 81
activity-level organizations, 70–71, 77
adaptation, 103, 105, 113
adopting Scrum
aspects, 3
assessing changes, 5–7
caveats, 7–8
effort required, 5
financial services, 32–36
first month, 13–15
impediments, 17
independent software vendors, 37–41
kickoff meeting, 11
process for, 9–11
reasons, 5
second month, 15–17
third month and beyond, 18–19
value-added card products, 29–32
Adventure Works, 4
Agile Project Management with Scrum
(Schwaber), 86
Agile techniques, 117–118
API (application programming interface), 92
architectures
decomposing, 47
integrating multiple-layer systems, 63–66
optimizing, 55–57
waterfall process, 22
auditing project teams, 125
automating operations, 52–54

B
bandwidth communication, 136–137
baseline plans
burn-down chart, 48–49
managing value, 87
purpose, 4, 106
bench, 73, 79
bidding work, 133–134
brainstorming impediments, 16
bugs, development velocity and, 96
burn-down chart
of baseline plans, 48–49, 87
Product Backlog, 109–111, 114
Sprint Backlog, 115
tracking progress, 124

C
change
assessing, 5–7
conflict and, 6
enterprises in transition, 29–41
functionality and, 90–91
in job responsibilities, 6–7
muscle memories hindering, 21–27
prioritizing, 17, 54–55, 73
code review in process control, 103
commitments, 25, 86–89
communication bandwidth, 136–137
compensation policies, 6, 14, 81
complexity assessment graph, 104
conflict, 6, 16–17
conflict resolution, 76
core functionality, 90–97
Corps Business (Freedman), 135
costs
controlling projects via, 86
extrapolating, 5
infrastructure changes, 92–93
of off-the-cuff requests, 19
predicting, 94
cross-functional teams
defined, 113
forming, 142–143
responsibilities of, 22, 106
waterfall process and, 143–145
workload management and, 135
culture. See enterprise culture
CVS software package, 74

D
Daily Scrums
defined, 113
distributed team membership, 82
initiating Scrum, 10
offshore development and, 138
organizing enterprise work, 72
in Scrum flow, 107–108
ScrumMaster, 15
team formation, 75
team room, 124
transparency and, 117
defects, development velocity and, 96
defined process control, 102
delivery dates, 86, 94
dependencies, external, 62, 83
development teams

collaboration on, 132
core functionality, 90–92
creating, 73
defined, 115
identifying impediments, 17
initiating Scrum, 9–11
managing people, 78–79
organizing for enterprise work, 70–73
product management and, 85–97
Transition Project Backlog, 15
development velocity
core functionality, 91
dead cores and, 94–96
defined, 39, 86, 117
documentation
core modifications and, 91
ingineering practices, 60, 68
for functionality, 112
organizational practices, 54
process development, 16
waterfall process, 22–23

e  
empirical process control, 26, 102–103, 114
engineering practices
integrating multiple-layer systems, 63–66
integrating work of teams, 66–68
overview, 59
work organized by functionality, 60–62
enterprise culture
changing, 4–7, 18–19
command and control, 23–24
extra managers and, 81–82
hiding reality, 26–27
impossible demands, 24–25
initiating Scrum, 9
muscle memory hindering change, 21–27
transitions, 29–41
waterfall process, 16
ETC (Enterprise Transition team)
defined, 9, 120–121
initiating Scrum, 9–11
noting impediments, 15
organization chart, 10
Sprint Planning Meeting, 13–14
external dependencies, 62, 83

F  
Fields, Mark, 15
financial products/services, 32–36, 51–52
finished product, 114
The Five Dysfunctions of a Team (Lencioni), 10
Ford Motor Company, 15
formung step (teams), 75
Forrester Research, 129
Freedman, David H., 135
functional specialization, 22
functionality
controlling projects via, 86
core, 90–97
determining ROI, 87
documenting, 112
extrapolating costs, 5
hiding reality and, 26
increments of, 112
infrastructure changes and, 90–93
integrating multiple-layer systems, 63–66
quality and, 90
team responsibilities, 106
for variable costs, 130–131
waterfall process and, 22
worked organized by, 60–62

G  
gMail, 80
Google, 80

I  
impediments
adopting Scrum, 17
decisions regarding, 4
management responsibilities, 7
noting in TPB, 15–16
off-the-cuff requests, 18–19
sources, 16–17
transparency of, 18
incremental development
defined, 114
of product functionality, 112
in Scrum process, 127–128, 130
transparency and, 117
independent software vendor (ISV), 37–41
infrastructure teams, 57, 60–62
inspection, 103, 105, 114
insurance industry, 97
integrating multiple-layer systems, 63–66
inventory reduction, 129
ISV (independent software vendor), 37–41
iterative development
bandwidth communication and, 137
defined, 114
enterprise change and, 27
ETC Scrum team, 9–10
workload management, 134
K
Kalz, Joris, 4
kickoff meetings, 11, 119–121
Krukal, Charles C., 135

L
Lean Thinking practices, 31, 36
Lencioni, Patrick, 10
lying to customers, 26–27

M
maintenance of dead cores, 96
management
command and control, 23–24
initiating Scrum, 9
people practices, 76–80
product, 6, 85–97
responsibility changes, 7
turnover in, 7
workload, 127–129, 134–136
managing value, 86–89
metrics, establishing, 14, 19, 124
muscle memory hindering change, 21–27

N
norming step (teams), 75–76

O
offshore development, 132–133, 136–139
Ogunnaike, Babatunde A., 102
Open Source movement, 141
operations, automating, 52–54
organizational culture. See enterprise culture
organizational practices
automating operations, 52–54
creating Product Backlog, 45–46
high-technology product companies, 46–51
optimizing architectures, 55–57
organizing enterprise work, 46, 51–52
organizing multiple views, 54–55

P
pace of work, sustainable, 4, 39
people practices
compensation, 81
distributed team membership, 82–83
extra managers and, 81–82
functional expertise, 80–81
managing people, 76–80
need for scarce skills, 83–84
organizing for enterprise work, 70–73
overview, 69
team creation, 73–75
team work, 75–76
people skills
functional expertise, 80–81
improving, 3
infrastructure changes and, 92
iterative development and, 106
need for scarce skills, 83–84
software development and, 103–104
performance considerations, 6, 79, 81
performing step (teams), 75–76
points of work, 48
preconditions, project, 14, 123–124
prioritizing changes, 17, 54–55, 73
process control
adaptation requirements, 103, 105
caveats, 7–8
defined, 102–103
empirical, 26, 102–103, 114
inspection requirements, 103, 105
visibility requirements, 105
process flow
adoption process diagram, 14
bandwidth communication and, 136–137
initiating Scrum, 9
iterative, 105
managing people, 76–80
overview, 106–108
roles, 106
Product Backlogs
addressing, 123–125
automating operations, 53–54
core functionality, 92
creating, 45–46
defined, 114
defining, 22
determining relative value, 87–89
development velocity, 86
establishing, 14
functional expertise, 81
functionality and, 5
grooming, 114
for high-technology product companies, 46–51
initiating Scrum, 9–10
integrating multiple-layer systems, 64–65
integrating work of teams, 66, 68
managing people, 76, 78
need for scarce skills, 83
optimizing product family architectures, 55–57
organizing enterprise work, 46, 51–52
organizing multiple views, 54–55
organizing people for enterprise work, 70–73
product development

Product Backlogs, continued
overview, 109–110
relative value, 87–89
in Scrum flow, 106–107
self-managing teams, 24
Sprint Backlog, 111–112
team creation, 73
team formation, 75
tracking progress, 124
value-driven development, 129
work organized by functionality, 61–62
product development
adopting Scrum, 3
complex software development, 103–105
conflicts, 16–17
team formation, 75
enterprise culture and, 4
integrating multiple-layer systems, 63–66
integrating work of teams, 66–68
optimizing product family architectures, 55–57
overview, 101–102
problem exposure and, 3
waterfall process, 16
product management, 6, 85–97
Product Owner
collaboration with, 134
core functionality, 92
defined, 106, 114
development team and, 85
distributed team membership, 82
functional expertise, 81
hiding reality from, 26
identifying, 14
identifying impediments, 17
increments of product functionality, 112
initiating Scrum, 9–10
integrating multiple-layer systems, 63–64
integrating work of teams, 66–67
job changes, 6
managing people, 76–80
managing value, 86–89
need for scarce skills, 83
organizing enterprise work, 51–53, 57, 71
Product Backlog, 109–110
product managers as, 6
removing, 79
responsibilities, 62
ROI responsibility, 47, 78, 86
in Scrum flow, 106–108
team creation, 73–74
training, 123
value-driven development, 127–129
work organized by functionality, 61
productivity
infrastructure development, 92–93
in iterative development, 106
managing, 79
as metric, 124
quality control and, 6
via self-management, 23–24
product-level organizations, 70–72, 78
programs. See projects
Project Goal, 114
Project Management Institute, 21
project teams. See teams
projects
auditing teams, 125
controlling, 86
defined, 114
establishing preconditions, 14, 123–124
identifying, 14
kickoff meetings, 11, 119–121
quality control, 86
realizing benefits of, 129–130
tracking, 124
trying out Scrum, 5
waterfall process, 16
prototypes, 18–19, 68
Q
quality control
cross-functional teams, 22
dead cores and, 94–95
functionality and, 90
muscle memory and, 21
projects and, 86
in Sprints, 6
R
Ray, W. Harmon, 102
relative value of Product Backlogs, 87–89
requirements
defined, 114
development velocity, 39
iterative development and, 106
parsing, 140
prevalence of changes in, 94
software development and, 103–104
waterfall process, 16, 22
responsibility, compensation and, 81
retrospective, 114–115
ROI (return on investment)
functionality and, 87
maximizing, 51
as metric, 124
Product Owner responsibility, 47, 78, 86
team creation and, 73
roles, 6, 106
rollout teams
composition of, 17
initiating Scrum, 9–11
pressuring, 17
Sprint Planning Meeting, 13–14
Transition Project Backlog, 15–16

S
Schwaber, Ken, 86
Scrum, 115
Scrum Alliance, 5
Scrum Center, 14, 125
Scrum development teams. See development teams
Scrum rollout teams. See rollout teams
Scrum teams. See teams
Scrumeteria, 32
ScrumMaster
defined, 106, 115
functional expertise, 81
identifying, 14
initiating Scrum, 9–10
integrating multiple-layer systems, 64
job changes, 6
managing people, 76–80
organizing enterprise work, 72
quality control, 6
removing, 79
reporting impediments to, 17
responsibilities, 15, 125
Scrum Center, 125
self-managing teams, 23–24
team creation, 73
training, 5, 123, 141
work organized by functionality, 60–61
self-managing teams
defined, 106, 113, 115
extra managers and, 81–82
management responsibilities, 7
muscle memory and, 21
need for scarce skills, 83
people practices, 76–80
productivity via, 23–24
workload management and, 135
self-organizing teams, 106
software development, 55–57, 103–105
SourceForge, 141
Sprint Backlog, 107, 111–112, 115
Sprint Goal, 115
Sprint Planning Meeting
coloring first, 13–15
defined, 115–116
defining Product Backlog, 22
distributed team membership, 82
extra managers, 81–82
handling external dependencies, 62
initiating Scrum, 10–11
managing people, 78
need for scarce skills, 83
participants, 13
in Scrum flow, 107–108
Sprint Backlog, 111
trying out Scrum, 5
virtual teams and, 142
Sprint Retrospective Meeting, 108
Sprint Review
core functionality, 92
defined, 116
distributed team membership, 82–83
engineering practices, 61
initiating Scrum, 11
integrating multiple-layer systems, 63–64
off-the-cuff requests, 19
in Scrum flow, 107–108
tracking progress, 124
virtual teams and, 142
Sprints
defined, 107, 115
definition of “done,” 113
distributed team membership, 82
increments of product functionality, 112
initiating Scrum, 9–11
integrating work, 79
quality control, 6
reviewing scope, 17
team formation, 75
work organized by functionality, 61–62
staff turnover, 6
stakeholders
defined, 116
initiating Scrum, 9
iterative development, 105
in Scrum flow, 108
storming step (teams), 75–76
sustainable pace of work, 4, 39

T
Team Formation Model, 75
teams. See also people practices; specific types of teams
adopting Scrum, 3
auditing, 125
coaallocating, 137
compensation policies, 6
composition of, 17, 24
creating, 73–75
Daily Scrums, 107–108, 124
dead cores and, 94
defined, 115
teams, continued

distributed membership, 82–83
Enterprise Transition teams, 9, 120–121
establishing project preconditions, 123–124
formation activities, 75–76, 123
functional expertise, 81
identifying, 14
improving, 3
infrastructure, 57, 60–62
initiating Scrum, 9–10
integrating work of, 66–68
iterative development, 105–106
job changes, 6
need for scarce skills, 83–84
problem solving in, 6
in Scrum flow, 107
size of, 137, 139–140
Sprint Backlog, 111
virtual, 140–142
technology, development and, 103–104, 106
testing process, 22, 92
ThoughtWorks, 129–130
“three block war,” 135
3M, 80
TPB (Transition Product Backlog)
initiating Scrum, 10–11
noting impediments, 15–16
posting, 17
sources of impediments, 16–17
Sprint Planning Meeting, 14

training
in first month, 13
management responsibilities, 7
Product Backlog and, 51–52
Product Owner, 123
in Scrum Center, 125
ScrumMaster, 5, 123, 141
teams, 76
trying out Scrum, 5
via Scrum Alliance, 5
TransCanada Pipelines (TCPL), 130
Transition Product Backlog. See TPB (Transition Product Backlog)
transparency in Scrum process, 18, 26, 103, 116
Tuckman, Bruce, 75

U
U.S. Marine Corps (USMC), 135

V
value, managing, 86–89
value-driven development, 86–89, 127–129
velocity. See development velocity
Virtual Scrum, 142
virtual teams, 140–142
visibility requirements in process control, 105

W
waterfall process, 16, 21–23, 143–145
workload management, 127–129, 134–136
About the Author

Ken Schwaber co-developed Scrum with Jeff Sutherland sixteen years ago. Since then he has run his own software company using Scrum and helped many others use Scrum. He is a signatory of the Agile Manifesto, and founder of the Agile Alliance and Scrum Alliance. Ken has been in the software business for over 35 years. He lives in Lexington, Massachusetts.