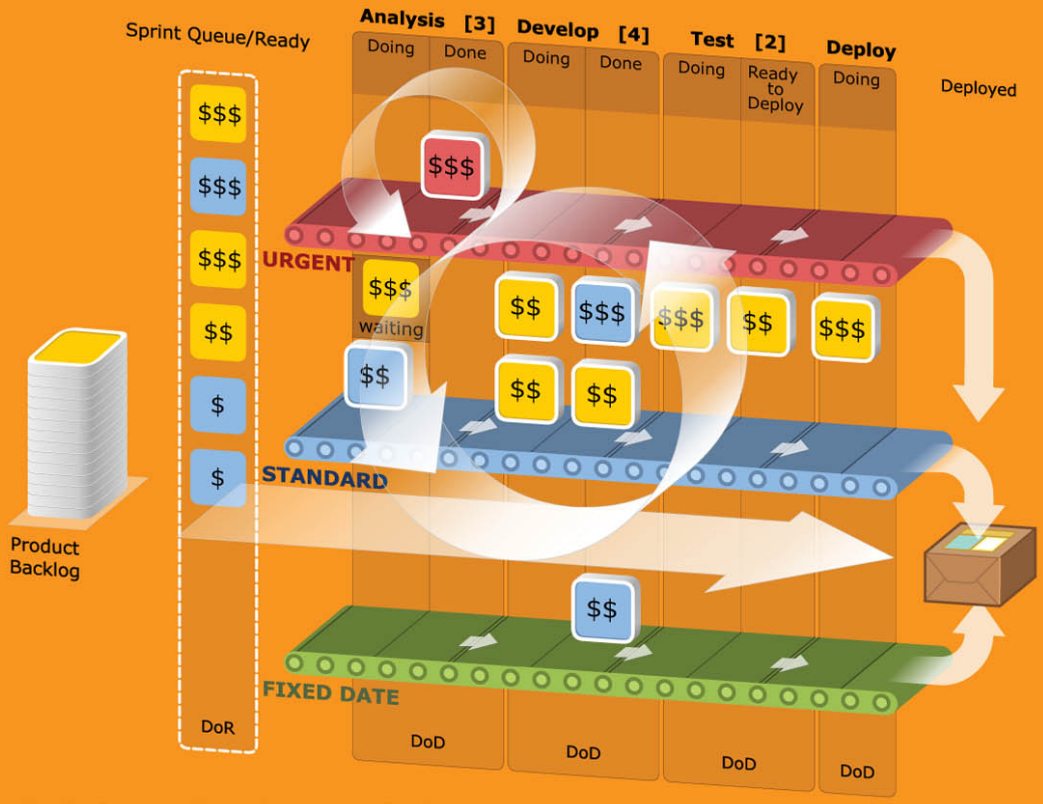
The Scrumban [R]Evolution

Getting the Most Out of Agile, Scrum, and Lean Kanban

Ajay Reddy

*Forewords by **David Anderson** and **Jim Benson***



Agile Software Development Series

Alistair Cockburn and Jim Highsmith, Series Editors

The Scrumban [R]Evolution

The Agile Software Development Series

Alistair Cockburn and Jim Highsmith, Series Editors

Addison-Wesley

Visit **informit.com/agileseries** for a complete list of available publications.

Agile software development centers on four values, which are identified in the Agile Alliance's Manifesto*:

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

The development of Agile software requires innovation and responsiveness, based on generating and sharing knowledge within a development team and with the customer. Agile software developers draw on the strengths of customers, users, and developers to find just enough process to balance quality and agility.

The books in The Agile Software Development Series focus on sharing the experiences of such Agile developers. Individual books address individual techniques (such as Use Cases), group techniques (such as collaborative decision making), and proven solutions to different problems from a variety of organizational cultures. The result is a core of Agile best practices that will enrich your experiences and improve your work.

The Scrumban [R]Evolution

Getting the Most Out of Agile, Scrum, and Lean Kanban

Ajay Reddy

Addison-Wesley
New York • Boston • Indianapolis • San Francisco
Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Library of Congress Cataloging-in-Publication Data
Reddy, Ajay.
The Scrumban [r]evolution : getting the most out of Agile, Scrum, and lean Kanban / Ajay Reddy.
pages cm
Includes index.
ISBN 978-0-13-408621-7 (pbk. : alk. paper)—ISBN 0-13-408621-X (pbk. : alk. paper)
1. Project management—Data processing. 2. Lean manufacturing. 3. Agile software development. 4. Scrum (Computer software development) I. Title. II. Title: Scrumban evolution. III. Title: scrumban revolution.
T56.8.R43 2016
005.1068'4—dc23

2015016226

ISBN-13: 978-0-13-408621-7
ISBN-10: 0-13-408621-X

Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana.
First printing, July 2015

I dedicate this work to my first born, Nehemiah Subhinay, named after Nehemiah 400–450 B.C., who ran the first successful Agile project in history, completing the wall around Jerusalem in 52 days.

This page intentionally left blank

CONTENTS

This page intentionally left blank

FOREWORD
BY DAVID J. ANDERSON

"Scrum is hard," says Ken Schwaber, one of the creators of the method. It's a defined process with a prescriptive set of rules that are known to work in combination. Scrum is known to work in the context for which it was designed. The real challenge for those adopting Scrum is in changing their context to enable Scrum to be effective for them. Changing the context of your business and the demands of your customers isn't easy—and hence, "Scrum is hard."

Back in 2002, when I was first contemplating what we now know as the Kanban Method, Scrum was not the dominant Agile method that it is today. Agile was new and the term "Agile software development" had been around for only one year. The best known of the Agile methods was Extreme Programming. Many software development organizations were still practicing methods developed in the 1980s and 1990s and using techniques such as use cases to capture requirements. Project planning was done with Gantt charts, and phase-gate governance processes that forced entire large-scale projects known as "phases" were commonplace. There would be a requirements gathering phase, a systems analysis phase, a design phase, and so on. Agile methods, with their small deliverable increments and iterative approaches to coding and testing, were scary and adoption was slow. Where I saw the greatest resistance was in adoption of looser approaches to requirements definition, such as the use of user stories, and in the dismantling of phase-gate governance processes often controlled by program/portfolio management offices or an audit or quality function often reporting into the PMO.

I came to the conclusion that if we were to deliver true business agility for the masses of technology development companies around the world, we had to adopt a new approach to improvement. Rather than trying to replace existing methods with new (but scary) Agile methods, we should instead pursue an evolutionary approach where we "start with what you do now" and look for specific problems that exist, and then seek specific ways to fix those.

By 2004, I saw a pattern where IT departments and product development groups would commit to too much work, too early; make plans often based on idealistic assumptions; and then fail to deliver on their promises, leading to unhappy customers. The causes of the failures formed a common pattern. Committed work would change in scope and definition, while new work would arrive, and the ongoing work would be interrupted often for information gathering reasons, or for rework, or for servicing production problems. Committed work in progress would get put aside and blocked for many reasons, and insufficient risk management planning had taken

place to allow for these problems. The solution for many of these problems was to adopt a kanban system as a first step in a process of evolutionary improvement. Kanban systems encourage us to defer commitment, limiting work in progress and limiting our exposure to many of the common problems that cause promises to be broken and projects to fail to deliver on expectations.

These two things were separate: We needed an evolutionary approach to improving service delivery in IT work and product development, and kanban systems were a good solution to unrealistic early commitment and the problems of interruptions, changing and growing scope, and work blocking for a multitude of reasons. It turned out that the pattern that required use of a kanban system was so common that by 2007 the concepts had effectively merged into one single management approach we now know as the "Kanban Method."

So Kanban was "agility for the rest of us." Kanban was designed to bring agility to those who couldn't or wouldn't adopt defined Agile methods in the 2000s. What does this have to do with Scrum? Well, Scrum is hard. It has probably been oversold to an eager audience keen to find a simple solution to their delivery challenges. Many firms have adopted Scrum without fulfilling its full requirements—they haven't changed their context to one that's truly suitable for Scrum. Indeed, in many cases, it simply isn't possible to control the context of your business; instead, your customers and your market do that for you. They don't change to fit the way you want to work; rather, you must change to fit the way in which they expect you to do business with them. This has left a world full of troubled, incomplete, or lackluster Scrum implementations.

It turns out that despite the smaller increments of sprints, many Scrum implementations suffer from all the same ailments that I observed with traditional project management and software development life-cycle methods more than a decade ago. Hence, adopting Kanban is an appropriate way to help a challenged Scrum deployment evolve and adjust to the context in which it is being used. This doesn't make the purists happy. It's not "Scrum by the defined set of rules," so, according to Ken Schwaber, it isn't Scrum. The question you have to ask yourself is, "Is our goal to do Scrum perfectly and be first-class citizens in the worldwide Scrum community? Or is our goal to improve the service delivery of IT work, software development, and new products to our customers?" If you recognize the latter as your true goal, then you have to accept that some degree of tailoring is required to develop a process that is unique to your own situation and "fit for purpose." The fact that you've picked up this book and taken the trouble to read the foreword would suggest that you already recognize that strictly following the rules of Scrum isn't working for you, and you need help to evolve to something that better matches the context of your business.

Ajay Reddy is an experienced Scrum practitioner who has been through the same journey you find yourself on. He, too, discovered that he needed help, and he found that the message of the Kanban Method resonated with him and enabled him to deliver solutions to what had appeared to be intractable problems. If Scrum hasn't been working for you, maybe the issue doesn't lie with you and your ability to follow

its strict rules properly. Maybe the truth is that you need a different process solution, uniquely tailored to your situation. Let Kanban help you achieve that, and allow Ajay to guide you in how to do it.

If there is a single message I want you to take away from this foreword, it is contained in this quote from my own book, *Kanban: Successful Evolutionary Change for Your Technology Business*: "You have permission to try Kanban. You have permission to modify your process. You have permission to be different. Your situation is unique and you deserve to develop a unique process definition tailored and optimized to your domain, your value stream, the risks that you manage, the skills of your team, and the demands of your customers." Be proud of what you've achieved so far with Scrum, and embrace the idea that there is so much more you can do in future. Let Kanban help you get there.

—David J. Anderson
Sequim, Washington

This page intentionally left blank

FOREWORD
BY JIM BENSON

If there was one central tenet of Lean, one that no one ever talks about, it would simply be this:

You have to pay attention to your project.

That is simply it. To continuously improve or to deliver a quality product, we need to actually know, as a team, what we are doing and why.

When we abandon our projects or our understanding of our customers to prescriptive management techniques, we are saying that we (our team and our product) are just another run-of-the-mill software factory. Predictable, mechanical, uncreative. Certifiable.

But very few projects can claim to have reached this level of the mundane. Software creation gains profitability only because of differentiators. Differentiators require vision. Vision requires creativity. And creativity requires experimentation and imagination.

We are not robots. Software is not an assembly line—far from it. Therefore, software developers and those drawn to the industry have some key shared characteristics:

1. They are problem solvers.
2. They are inventors.
3. They are craftspeople.
4. They get really angry when they are denied to be 1–3.

In 2008, when my colleague Corey Ladas wrote *Scrumban: Essays on Kanban Systems for Lean Software Development*, he was extending the definition of increasingly prescriptive Agile techniques by providing two important things:

- A toolkit to pay attention
- Permission to pay attention

Software development is a system that is creating other systems. It is intricate. It is social. It is complex. We don't have shelves and shelves of books on how to develop software because it is easy. Each project has many potential implementation paths, many potential life cycles, and many decisions.

In turn, each website, app, or embedded program we create is itself a system. It interacts with the user, it interacts with other software, it breaks, it needs to be maintained. Our software products live within social, financial, and digital worlds that we must at least appreciate, if not understand, to craft appropriate code.

So I must honestly ask of you and your current Agile practice:

- Are you setting your development teams up with a system that allows them to really see the real-time context of their work?
- Does your system allow your team to notice early when there are problems?
- Does it allow them to change not just the features, but also your processes?
- Does it really allow them a sustainable pace (develop a quality product) or are you always trying to improve your velocity (increase utilization)?

To do any of these things, we cannot afford to separate the developers from the design. Current versions of popular Agile techniques incorporate mechanisms to divide developers from the vision of their product. Our original goals with Agile were to liberate the programmers. Instead, we have come full circle. Scrum masters have become project managers; product owners have become product managers. We have recreated waterfall in Scrum clothing.

The Kanban Method, Personal Kanban, Scrum (Types A, B, C), Scrumban, SAFe, RUP, XP . . . none of these things is THE answer. There is no one, single answer. If there was, it would be obvious and we'd all just do it. Taken alone, these approaches are as healthy as a prepackaged microwave dinner. When these and other techniques are used as ingredients, then you can make a real meal.

There would be no Kanban today without Kent Beck, Eliyahu Goldratt, Taiichi Ohno, and W. Edwards Deming. David's, Corey's, and my experiences in launching Kanban were themselves the products of learning and paying attention—of combining many ideas into one.

Just as a driver's manual from Tokyo won't tell you precisely how to drive in Akron, so the methods and ideas mentioned earlier won't tell you precisely how to steer your projects if they are used in isolation. To steer, you have to pay attention. To pay attention, you need to see your work and reduce distractions.

Lean and Kanban will *help* you do these things, but they are frustrating.

Why?

Because you and your team are likely overworked and don't have the time or perhaps even the permission to pay attention to or change your working environment. Open-ended Lean systems are therefore hard to start. You need to have clear direction and descriptions of work for your team members, your bosses, and other stakeholders. *Who has time for that?*

That's where this book comes in.

Ajay is taking Corey's ideas a step further, providing some more potential implementations for teams who would rather be successful than blind followers of canned

processes. This book will help you to pay attention, to see the work you are taking on, to demonstrate to others the load you are dealing with, and to begin to take control of your work in process. *But don't stop here: Ajay calls it (r)evolution on purpose. Evolving is a continuous process.*

Think of this book as that first glimmer of light at the end of the tunnel. But remember, simply seeing that light is not enough: You need to keep moving forward, keep improving your processes, and keep creating and maintaining the best product possible.

Software never sleeps.

—Jim Benson
Author of the Shingo Award-winning *Personal Kanban: Mapping Work | Navigating Life*
Ocean Shores, Washington

This page intentionally left blank

PREFACE

Although Scrumban has evolved as a framework over the years, it has no definitive guide or definition. In fact, as highlighted early in this book, several "authoritative" sources disagree about what Scrumban actually represents. Scrumban deserves better.

The objective of this book is to transform a confusing set of conflicting representations into a comprehensive, coherent, and practical resource that brings value to the individuals and organizations wishing to harness the power of Scrumban in their own environments. This book is not about new advances in thinking, but rather about demonstrating how a broad set of proven Lean and Agile principles can be effectively interwoven and managed within the context of a single framework. In the same vein, it's not a recipe book. Although it incorporates recommendations and practice tips, this book is primarily intended as a guide to allow individuals and organizations to expand their thinking in ways that effectively support their ultimate aims.

How to Use This Book

There are a couple of different strategies for approaching this book. Before addressing them, let's consider the different levels of learners.

New practitioners want to seek out concrete steps to implement and follow. Scrumban presents a couple of unique challenges for these "Shu-level" learners. First, it calls for practitioners to engage in systems thinking and to acknowledge the notion that we can't improve the systems of which we're a part without first gaining a holistic understanding of those systems. Second, though some of Scrumban's practices and principles lend themselves to defining concrete steps, most do not. Consequently, the concrete steps early learners will be asked to embrace represent incremental building blocks to larger understandings.

Now for the two strategies. Which one you select depends on what you want to get out of this book:

- If you're interested in understanding the big picture and want to master Scrumban in its full context (something I strongly encourage), then moving through this book from beginning to end is the way to go. Should you find yourself reading something that you think is irrelevant or puts you to sleep, then by all means skip ahead. But I don't suggest doing this too often—gaining a comprehensive understanding of the full context is especially important to helping new practitioners avoid common pitfalls.

- If you're interested in learning more about a specific topic, you can jump straight to the chapter that covers it. Each chapter is a self-contained vehicle of knowledge on a broad topic, but the chapters are also ordered to lend a sense of continuity to the journey that is Scrumban.

Throughout the book I've sprinkled stories about how teams and organizations have used Scrumban. In some cases, teams or organizations accelerated their mastery of Scrum's roles and practices. In other cases, Scrumban served as an alternative path to Agility (which meant the teams following this path were no longer practicing Scrum). Both kinds of outcomes delivered pragmatic, bottom-line results—the thing teams and organizations care most about. If you don't care about these things, then this book is not for you.

Although this book predominately focuses on software development, the Scrumban framework can be adopted across a variety of business functions, it facilitates shared languages and shared understandings, and it integrates and aligns efforts effectively. This aspect of the framework is particularly relevant to challenges associated with scaling Lean and Agile practices across the enterprise, and is what ultimately makes Scrumban so powerful.

How This Book Is Organized

This book is organized into four main sections, each consisting of several chapters.

- **Part I, Introduction**
  - **Chapter 1, Manifestations: Scrumban Demystified**
    An overview of Scrumban's origins and current states of understanding in Lean and Agile circles.
- **Part II, Foundations—Starting with the End in Mind**
  Chapters 2, 3, and 4 are intended to provide readers with a holistic overview of Scrumban's roots and the reasons to employ it. The holistic nature of these chapters makes the material relevant for all levels of learners (Shu, Ha, and Ri—I'll talk more about these levels in Chapter 1), but especially for managers and executives. Because deep foundational understandings at every level drive better outcomes, I encourage individuals at the team level to avoid giving these chapters short shrift—especially Chapter 2.
  - **Chapter 2, The Matrix and the Mess: Where It All Begins**
    This chapter introduces key considerations that often go unrecognized in quests for improvement. I visit historical factors that often influence the performance of IT organizations, explore why efforts to become Agile (which often involves introducing Scrum into an environment)

can stall or fail, and how Scrumban is generally structured to help overcome these impediments.

- **Chapter 3, The Mission: Clarifying the Relationship between Purpose, Values, and Performance**
  Organizations are most successful when they maximize the alignment of efforts across four key areas. In this chapter, I highlight how Scrumban can be used to improve organizational alignment across all business functions.
- **Chapter 4, Motivations: Why Scrumban Works**
  Why Scrumban? I take a closer look at the core principles and practices that make Scrumban such a robust framework for improving worker satisfaction and creating high performance.

- **Part III, Execution—Putting Scrumban into Practice**
  Chapters 5, 6, and 7 are where the rubber meets the road, emphasizing the "how to" versus the "why." Because I strongly discourage new practitioners from diving into the "how to" with little understanding of the "why," readers will still find a significant amount of explanatory material embedded within content that's intended to serve as a practical guide.
  - **Chapter 5, Mobilize: Rolling Out Scrumban**
    A close look at one approach for kickstarting Scrumban within a team or organization.
  - **Chapter 6, Method: Working under the Hood**
    A concrete exploration of how Scrumban's specific values, principles, and methods can be applied and used at different stages of adoption.
  - **Chapter 7, Measurements: Gaining Insights and Tracking Progress**
    The metrics I rely on to provide new perspective and enable more reliable forecasting.
- **Part IV, Improving—Advanced Topics and Practices**
  Chapters 8, 9, and 10 represent advanced thinking—a "graduate"-level course. Unseasoned practitioners are less likely to fully understand the relevance and power of the topics discussed, as that understanding comes only with experiencing Scrumban in action. Nonetheless, the content provides perspective on the many options and capabilities the framework exposes over time.
  - **Chapter 8, Management: Management Is Doing Things Right—Leadership Is Doing the Right Things**
    Though Scrumban doesn't have to be driven from a top-down direction to achieve initial success, this chapter reviews why good leadership is essential to sustaining high performance over the long term.

- **Chapter 9, Maturing: Like a Fine Wine, Scrumban Can Get Better with Age**
  Approaches that help teams continue to mature, and common evolutions that might be encountered.
- **Chapter 10, Modeling: To Boldly Go Where Few Men and Women Have Gone Before**
  Even more advanced techniques for more precise forecasting and adaptive management.
- **Appendix, More: For the Stout of Heart**
  An appendix containing definitions and additional reference material.

Conventions

I've employed a couple of simple conventions to help readers navigate through certain categories of information. For example, some topics are especially suited to executives and those in portfolio or program management roles. You'll encounter illustrations of executive and manager types introducing these topics (I've borrowed these illustrations from GetScrumban, an online game I co-created as a training tool).

Similarly, "Sanjay" the coach will direct your attention to concepts of particular importance, call out traps and pitfalls for the unwary, and otherwise offer tips based on experiences helping many teams leverage Scrumban in their environments.

I also try to let you know when more advanced topics are addressed so you can elect to revisit them as your understanding grows over time.

Acknowledgments

This work would not have been possible without the contributions and support of many people. In particular, I would like to thank. . . .

. . . my dear wife Diana, our four children, and my mother Susheela for their exuberant and selfless support to making this work possible.

. . . my friend and colleague at CodeGenesys, Jack Speranza, for his help in making this book possible. As CodeGenesys Chief Operating Officer, he provided a pedagogical perspective in presenting this book's material to a variety of audiences.

. . . Marc Hughes, my friend and partner, and my team at ScrumDo for supporting this work with invaluable data and analysis.

. . . clients of CodeGenesys and ScrumDo.com who gave us permission to share their stories.

. . . other colleagues at CodeGenesys—especially Haritha and Bernard—for their support and help.

. . . Dimitar Bakardzhiev and Frank Vega for their invaluable feedback.

. . . David Anderson, Jim Benson, and Russell Healy for their kind support.

. . . Corey Ladas, creator of Scrumban, without which this book obviously could not exist.

. . . and the too-many-to-name members of the Lean and Agile communities who agreed to let us incorporate portions of their work and are always making many contributions for the greater good.

—Ajay Reddy
Boston, Massachusetts

This page intentionally left blank

About the Author

Ajay Reddy has been helping technology teams and organizations improve how they work for more than a decade. Emphasis on a hands-on approach and collaborative experimentation, verified business outcomes, and improved team satisfaction are a hallmark of his engagements.

Ajay began his professional journey as a software engineer, a role in which he was first exposed to Agile approaches like Extreme Programming. He soon transitioned to coaching others in adopting Agile mindsets and practices, constantly seeking out ways to improve this process based on the variety of challenges associated with different settings.

Ajay founded CodeGenesys, a Lean–Agile consulting boutique, in 2009. In 2010, he co-developed ScrumDo, a Scrum and Kanban management tool, with the express purpose of facilitating Agile implementations across the industry. Over the past five years, he has helped organizations both large and small employ Scrumban with great success. Ajay believes Scrumban is a particularly simple, yet powerful framework with a great amount of untapped potential. In addition to writing this book, he recently co-created the GetScrumban game as a practical and effective aid for helping individuals and organizations orient themselves to the true capabilities of this framework.

As the chief product strategist and lead coach for ScrumDo, Ajay is helping teams and organizations in 145 countries realize the benefits of Scrumban. He teaches Scrumban across the United States, India, and Europe, and is a regular speaker at Scrum and Kanban meetups and conferences across the globe.

In 2012, Ajay became disillusioned with the debate over frameworks, seeing it as putting too much emphasis on the frameworks rather than on the human systems and business concerns those frameworks were created to facilitate. He believes using frameworks as tools to effectively support desired outcomes is more important than the mechanics of any framework. It was this realization that led directly to his spending the last two years writing this book.

Ajay Reddy lives in Massachusetts with his wife, Diana, and their four children. You can find him on Linkedin, at www.ajayreddy.net, at codegenesys.com, at www.scrumdo.com, and as @ajrdy on Twitter.

This page intentionally left blank

Chapter 1

Manifestations: Scrumban Demystified

I estimate that 75% of those organizations using Scrum will not succeed in getting the benefits that they hope for from it . . . Scrum is a very simple framework within which the "game" of complex product development is played. Scrum exposes every inadequacy or dysfunction within an organization's product and system development practices. The intention of Scrum is to make them transparent so the organization can fix them. Unfortunately, many organizations change Scrum to accommodate the inadequacies or dysfunctions instead of solving them.

—Ken Schwaber, co-creator of Scrum

Scrum is an incredibly simple, effective, and popular software development framework; its value increases as teams and organizations develop their understanding and application of its core principles and practices.

Despite its simplicity, Scrum can be difficult to master. While it empowers both individuals and teams to discover factors that impede Agility and address how they should be tackled, it relies on their collective motivation, effort, and capabilities to make this happen. An entire industry now exists around delivering services to help individuals, teams, and organizations rise to higher levels of capability.

Scrumban is also a simple framework. It's a relative newcomer to the world of software development and has yet to fully evolve; it's also often misunderstood by people in Lean and Agile circles. Many people have never even heard of it. Some believe it to be nothing more than using virtual kanban systems within the Scrum framework, while others believe it to be a new software development framework that combines "the best" elements of Scrum and the Kanban Method. Neither of these viewpoints captures the true essence of Scrumban.

A Helpful Perspective

In the early 2000s, Alistair Cockburn introduced "Shu-Ha-Ri" to the software development world. It provided a way of thinking about the three distinct phases people pass through as they master a new skill or concept. Cockburn borrowed this concept

from Japanese martial arts and believed it could be effectively applied within the context of improving software development processes and practices.

I invite you to embrace a learning style that is loosely based on Shu-Ha-Ri in learning to understand Scrumban. Let's quickly review the three stages:

- ***Shu (Beginner)*:** The first stage of learning. New learners seek to reproduce a given result by following a set of instructions that are practice focused. They focus on how to perform a task with a basic understanding of the underlying principles. Success in this stage is measured by whether a procedure works and how well the student understands why it works.
- ***Ha (Intermediate)*:** Once a student has mastered basic practices, values, and principles, he or she begins to identify the limits of these practices and techniques. The student seeks to expand his or her awareness of alternative approaches, learning when an alternative applies and when it breaks down. Success in this stage is measured by how well the student learns to apply adaptive capabilities to varying circumstances.
- ***Ri (Advanced)*:** The student has become a master. It no longer matters whether he or she is following a given procedure or practice. His or her knowledge and understanding are the product of repeated thoughts and actions. The master has developed a fully adaptive capability within the context of his or her experience in the environment. Success is measured by consistently successful outcomes.

Informed readers should not confuse the concept of Shu-Ha-Ri with something like Carnegie Mellon University's Capability Maturity Model Integration (CMMI) process improvement training and appraisal program. Those who adopt and practice Scrumban principles would not be as inclined to direct harsh criticisms at the model as have some individuals in Lean and Agile circles.

Although I disagree that CMMI's prescribed "destinations" or "characteristics" correlate with capability in all contexts, this model does present a menu of potential catalysts to employ in a Scrumban framework when pursuing desired business outcomes. Viewed from the opposite direction, the flow management capabilities that Scrumban enables may provide a useful catalyst for organizations pursuing prescribed levels of CMMI capability to achieve their desired outcomes.

A Framework for [R]Evolution

When Corey Ladas introduced the world to Scrumban in his seminal book, *Essays on Kanban Systems for Lean Software Development* (Modus Cooperandi Press, 2009), he defined Scrumban as a transition method for moving software development teams from Scrum to a more evolved development framework. Over the past five years, my

own research and work experience have unearthed the many ways in which Scrumban itself has evolved.

Since Corey wrote his book, organizations have layered the Kanban Method alongside Scrum to help them achieve several different kinds of outcomes. For example, I've successfully helped organizations apply Scrumban principles and practices in a variety of contexts—from startups to Fortune 50 companies, in industries ranging from professional services to software product development. Across these contexts, I've used Scrumban for the following purposes:

- Help teams and organizations accelerate their transitions to Scrum from other development methodologies
- Enable new capabilities within teams and organizations to help them overcome challenges that Scrum (purposefully) causes them to confront
- Help organizations evolve new Scrum-like processes and practices that work for them—not to accommodate the inadequacies and dysfunctions that Scrum exposed, but rather to resolve them in a manner that is most effective for their unique environment

These experiences demonstrate that Scrumban has evolved to become a family of principles and practices that create complementary tools and capabilities. And like any living organism, these principles and practices will continue to evolve as practitioners share their experiences and learnings.

Now, let's consider the three different outcomes summarized previously within the context of Shu-Ha-Ri:

- Scrumban provides the discipline and structure needed by practitioners in the Shu phase of learning. The Scrumban framework enables teams and organizations to manage the introduction of the artifacts and ceremonies of Scrum or the enhanced metrics and flow management practices of Kanban—disciplines and structures that new learners require in limited scope.

  For example, Scrum's ceremonies are essential to creating desired levels of performance and agility. Although it is a relatively simple framework, Scrum can seem overwhelming when it is first introduced. In a misguided effort to ease adoption, many organizations modify or omit ceremonies or, even worse, ignore the importance of understanding the basic principles and values. This rarely, if ever, produces the desired outcomes.

  Additional capabilities provided by the Scrumban framework can substitute for the functions served by the modified or omitted Scrum ceremonies. Scrumban's visualizations and other mechanics improve the effectiveness while reducing the time and effort associated with conducting ceremonies. Scrumban more effectively connects the practice of ceremonies with the principles and values that the ceremonies serve.

- Scrumban exposes new tools and capabilities that aid the experiments and discoveries pursued in the Ha phase. Meeting the challenges that teams and organizations commonly face as they implement Scrum or other Agile practices represents one aspect of this dimension.

  Consider creating and managing a product backlog. This Scrum artifact, and the events surrounding it (grooming and planning sessions), is intended to manage risk and optimize value by enabling better decision making due to maximized transparency and understanding of work. This can be especially frustrating when organizations effectively assign multiple product owners to a backlog because individual limitations interfere with realizing a full set of capabilities, or because of wide variations in subjective assessments.

  The Scrumban framework visualizes information and integrates capabilities other frameworks don't inherently provide. It helps provide improved contextual understandings and more accurately measures the outcome of different approaches (directly appealing to the Ha phase practice and understanding). For instance, Scrumban visualizes all sources of work demands and a more objective economic impact over time (cost of delay) to help prioritize work, lending greater transparency to the overall picture and expanding ways to adapt.
- Scrumban is flexible enough to provide Ri-level masters with a robust process within which to operate at hyper-performing levels. By emphasizing systems thinking, experimentation, and discovery, Ri-level masters are free to mold their ways of working in whatever fashion will produce the best results—from both performance and worker satisfaction standpoints. It makes no difference whether the resulting process is true to any particular set of practices.
- Scrumban supports both "revolution" and "evolution." More importantly, it's structured in a way that strongly supports all levels of learning and understanding—at a level of quality that is greater than that provided by either Scrum or Kanban alone.

All of Scrumban's added capabilities can be *optionally* applied to a Scrum context in a variety of ways. They can also be extended across multiple areas of an organization to drive better business outcomes. Scrum's software development framework lies at its foundation, as does the Kanban Method. However, neither of these frameworks represents a prescribed destination for organizations practicing Scrumban.

Beyond representing a significantly evolved mindset from the framework expressed by Ladas, today's Scrumban is quite different from the definitions used by other respected leaders in the Lean/Agile community.[1] In many respects, these perspectives view Scrumban as a vehicle for moving teams from Scrum to another software development process. While this remains *one* potential outcome, real-world

1. See, for example, http://tiny.cc/badcomparisonsSK (July 2013).

applications demonstrate Scrumban has come to entail more than this across a broad range of contexts.

Over the years, Scrumban has been used to help teams and organizations accelerate their transitions to Scrum from other development methodologies. It's been used to help teams and organizations overcome a variety of common challenges that Scrum is designed to force them to confront. When the context requires, it's been used to help organizations evolve new Scrum-like processes and practices that work best for them—not simply as a means to accommodate inadequacies and dysfunctions Scrum exposed, but rather as a strategy to resolve those problems in a manner that is most effective for that environment. This latter outcome is obviously not something for which Scrum itself provides. These different paths reflect Scrumban's bottom line—the service-oriented pragmatism that most businesses value.

Ultimately, Scrumban has become a framework of empowerment. David J. Anderson, pioneer of the Kanban Method, recently stated:

> *Empowerment isn't about letting people do whatever they want, or assuming they'll somehow self-organize to produce the right outcome. Empowerment is about defining boundaries, and we do the same with children when bringing them up; we tell them things like when their bedtime is, where they're allowed to play, whether they're allowed to go outside the yard of the house, they're allowed to swim at the shallow end of the pool, they aren't allowed to jump from the diving board . . . all these things. So empowerment is about giving people clear boundaries, and then letting them use their initiatives inside the boundaries.*[2]

Scrumban is distinct from Scrum because it emphasizes certain principles and practices that are quite different from Scrum's traditional foundation. These include the following:

- Recognizing the role of management (self-organization remains an objective, but within the context of specific boundaries)
- Enabling specialized teams and functions
- Applying explicit policies around ways of working
- Applying laws of flow and queuing theory

Scrumban is distinct from the Kanban Method in the following ways:

- It prescribes an underlying software development process framework (Scrum) as its core.
- It is organized around teams.
- It recognizes the value of time-boxed iterations when appropriate.
- It formalizes continuous improvement techniques within specific ceremonies.

2. http://tiny.cc/DavidAKanban (March 2013).

Stop Drinking the Kool-Aid

Mike Cohn, a leader in the Agile/Scrum community, recently "criticized"[3] Scrum teams for not being "focused on finding innovative solutions to the problems they [teams] are handed." He wasn't actually criticizing Scrum; rather, he was criticizing a mindset that has evolved among Scrum practitioners—a mindset that favors a safe approach to completing work over innovation.

I see a related phenomenon in my coaching engagements. The biggest impediment to improvement often lies with team members who are either unable or unwilling to think about improving the way work is done (or how their work is relevant to creating business value).

I believe the problem runs even deeper than this. A cult of Scrum has arisen that permeates the industry that has developed around Scrum. Not only are Scrum practitioners failing to pursue innovation in their work, but they are also failing to pursue innovation in the process. Scrum has evolved over the years as new information was discovered, yet there seems to be a growing resistance among its most ardent practitioners to reflecting on how to support its fundamental purpose.

I saw this reluctance most recently during a presentation to an Agile community in Boston. At the beginning of the presentation, I asked the audience how many of them were using Scrum in their environments. About half the audience members raised their hands. Then I asked how many had experienced challenges in adopting Scrum in their organizations. Not a single hand went down.

As I began describing some of the alternative ways Scrumban enables teams and organizations to achieve their desired purposes, debates erupted over whether prescribed or popular approaches associated with Scrum were ineffective. Despite my repeated emphasis that Scrumban simply represents *alternative* or *additional* paths when some aspect of the Scrum framework isn't fulfilling its intended purpose in a particular context, the majority of people in the room—even those who acknowledged challenges with their Scrum adoptions—were more interested in defending their existing process than in considering whether an alternative approach could help them overcome a challenge.

This cult-like mentality is not limited to the Scrum community. Pick your method or framework, and you'll find a lot of similar behavior—even in Kanban.

Fortunately, most day-to-day practitioners are pragmatists and realists. Simple pragmatism and a willingness to experiment are why Scrumban has evolved to become the framework described in this book. Unfortunately, there will always be a fringe set of "thought leaders" who perpetuate framework debates because they are unwilling to promote the benefits of an approach that "competes" with models in which they've invested significant amounts of both time and money. Scrumban may

3. http://tiny.cc/cohnscrumcricism.

be somewhat less threatening because of its familiar elements, but they will criticize it anyway.

Scrumban is a framework that almost forces its practitioners to accept the reality that good ideas can come from anywhere. It encourages people to actively pursue this reality at every turn. The question readers must answer for themselves is whether they're ready to accept this reality and embrace exploration, or whether they will remain trapped within a narrower perspective, justifying this mindset by the existence of a "debate" that is perpetuated more out of fear than out of fact.

Let's Get Started

One of the most powerful characteristics of Scrumban is the fact it can be implemented at any level of the organization—you don't need the authority of someone in command and control to begin making a difference in how you work (though it's certainly easier if you do have some degree of buy-in).[4] It also tends to be contagious.

So whether your goal is to have your company become a market leader or simply to gain better control over your own environment, I invite you to join the Scrumban community's Scrumban.io[5] LinkedIn group or www.facebook.com/Scrumban and to follow the Scrumban blog at www.scrumban.io.

4. This can be true even in environments where development processes are subject to audit, though the nature and extent of evolutionary change may be more limited than in less restrictive contexts.
5. The Linkedin group is available at www.theagilecollaborative.net, which redirects to https://www.linkedin.com/groups/Scrumbanio-7459316.

This page intentionally left blank

Chapter 5

MOBILIZE: ROLLING OUT SCRUMBAN

IN THIS CHAPTER

- How Framework Choices Influence Outcomes
- A Step-by-Step Guide to Getting Started
- Using Scrumban to Stabilize a Team before You Improve

Having outlined the foundational principles and mechanics that make Scrumban successful, it's time to discuss how teams and organizations can start employing Scrumban. Rolling out Scrumban doesn't have to require a lot of effort. In fact, the approach outlined in this chapter can be completed in a single day.

Your Starting Condition

There are essentially three potential starting conditions for rolling out Scrumban:

- You're working with a team or organization for which both Scrum and Scrumban represent new ways of working.
- You're working with an existing Scrum team or organization that will use Scrumban to improve its mastery of the responsibilities and ceremonies associated with the Scrum framework.
- You're working with an existing Scrum team or organization that will use Scrumban to monitor performance, diagnose issues, and adapt existing practices to the form best suited for their context.

It's best if all teams participating in a formal kickstart program share the same origin, but it won't necessarily be fatal if they don't.

When You're New to Scrum

Scrum can be difficult to master, even though it is a relatively simple framework. Recognizing this reality, we've made Scrumban our chosen framework for introducing Scrum to teams and organizations for the first time.

Because Scrumban seeks to minimize the disruptions associated with imposing new definitions and responsibilities upon employees, rolling out Scrum under this framework is substantially different from traditional approaches. Rather than starting out with Scrum-specific orientation and training, we emphasize discovery of existing systems and processes, then use the framework to gradually introduce elements of Scrum as warranted by the context.

This gradual introduction can be both role based and process based. For instance, the "daily scrum" is a process-based change the team can begin to employ within the context of a Scrumban framework, just as new Scrum masters can be eased into their responsibilities one element at a time.[1]

The Kickstart Process

Coaching Tip!

Prefer to bypass the background detail provided in this section? Check out our quick Kickstart reference in the Appendix.

The kickstart process covered here is particularly effective when you're faced with limited time or resources—in other words, when teams or organizations need to better manage workflow, but don't have sufficient resources to develop those better ways. Naturally, this is not the only way to introduce Scrumban to teams.[2] Always use your understanding of Scrumban principles to modify the process to fit your context.

Incidentally, this process is modeled after a Kanban kickstart process developed by Christophe Achouiantz, a Lean/Agile coach, and Johan Nordin, development support manager at Sandvik IT. Sandvik needed a way to empower its company's teams to begin a process of continuous improvement that wouldn't involve a significant initial investment of time or effort. Achouiantz and Nordin have documented this process and made it available to the public as a reference guide. While the particulars

1. In some contexts, the role of Scrum master may even remain optional. In working with a variety of organizations over the years, Frank Vega notes that one of his earliest efforts to help a team move toward more iterative and incremental ways of working dispensed with this role altogether. It remains one of the most effective teams he's ever worked with. I don't necessarily consider this an ideal approach, but mention it only to underscore the importance of contextual discovery and decision making.
2. For example, the approach taken by Siemens Health Services (see the Appendix for the full case study) would not have called for individual teams to engage in a process of systems discovery and definition.

may not apply directly to your needs, it's an outstanding summary of the key points and objectives relevant to any process.

Preparation

As systems thinkers, it's critical to first understand the context of the environment before attempting to kickstart anything. At a minimum, preliminary preparation should include the following steps:

- Identifying the current conditions and organizational priorities (e.g., increased quality, productivity, predictability)
- Developing working relationships with key managers and team leads
- Ascertaining areas of desired change
- Introducing Kanban/Scrumban as a framework for evolutionary change
- Starting to educate staff on Scrumban's core principles and practices
- Identifying any risks and potential impediments to introducing Scrumban to targeted teams

Regarding that last item, it's important to recognize that some teams or contexts may not benefit from introducing Scrumban until certain conditions or impediments are addressed. These can include teams in an active state of reorganization, teams with serious internal conflicts, and so forth. Your context will determine how you address these situations. For example, Sandvik's needs dictated that such teams be bypassed until circumstances changed.

Though it's possible for teams to initiate Scrumban adoption without organizational buy-in, as previous chapters indicate, broader engagement is necessary to achieve the full breadth of desired outcomes and to sustain them over time. As such, Scrumban is not substantially different from a pure Scrum context.

PMO/Program Managers:

If you're overseeing a pilot program or broad-based transformation effort, we strongly encourage you to consider the recommendations outlined here. The choices you make in the early stages will greatly influence your ultimate success.

In his book *Succeeding with Agile: Software Development Using Scrum* (Boston: Addison-Wesley, 2009), Mike Cohn addresses a variety of considerations that apply when you're selecting the project context in which to roll out a new process (these considerations are made with an eye toward building off demonstrated success). Though less critical in a context where the Scrumban or Kanban framework is employed, they nonetheless represent worthy

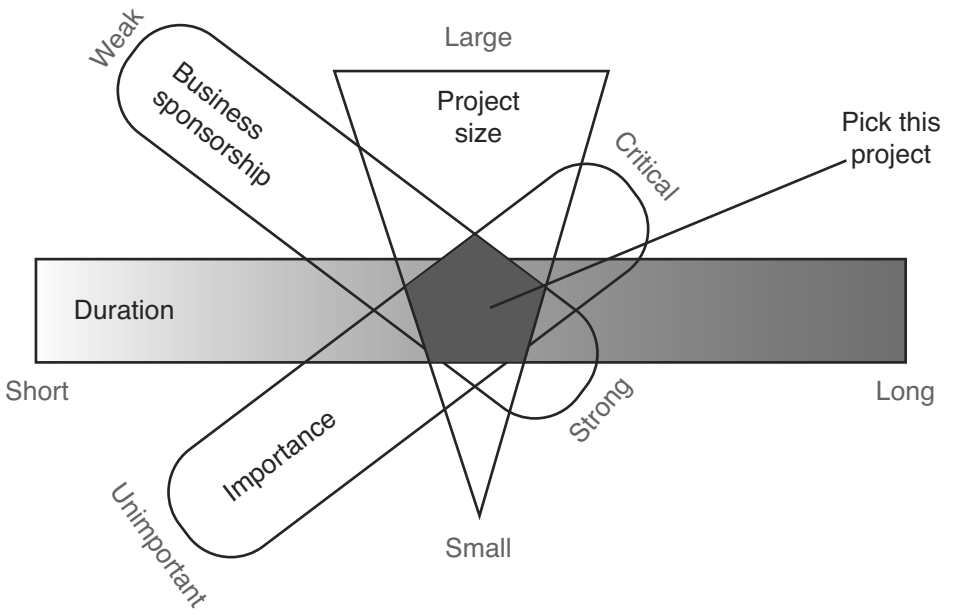


FIGURE 5.1 The convergence of key considerations in selecting a pilot project for introducing Scrum. These same considerations apply to Scrumban.

Source: Michael Cohn. *Succeeding with Agile: Software Development Using Scrum* (Boston: Addison-Wesley, 2009).

considerations (see Figure 5.1 for a graphical illustration of the convergence of these considerations):

- ***Project size***: For Scrum pilots, Mike suggests selecting a project that will not grow to more than five teams, and limiting initial efforts to just one or two of those teams. Coordinating work among more Scrum teams represents more work than you can effectively tackle.

  For Scrumban rollouts, there's substantially more leeway. With fewer concepts to learn and master (again, we start out respecting current roles and processes), working with a slightly larger number of teams is feasible.
- ***Project duration***: Select a pilot project of average duration for your organization, ideally around 3–4 months. This is plenty of time for teams to begin mastering the framework and seeing benefits. It's usually also sufficient for demonstrating that your new approach will lead to similar success in other contexts.
- ***Project importance***: Mike suggests that with Scrum pilots, it's critical to select high-profile projects. Unimportant projects will not get the organizational

attention necessary to make Scrum successful, and team members may be disinclined to do all the hard things Scrum requires.

There's slightly more leeway with Scrumban. Building momentum and trust through success is still an important objective, but the framework's service-oriented agenda and active management of psychological barriers to change represent additional capabilities that can be leveraged to ultimately satisfy these needs.

Executives:

Your efforts to educate your organization's nontechnical employees about how this undertaking is important to the business, and encouraging their active engagement in the process, will go a long way toward creating long-term success.

- *Business owner's level of engagement*: As previously mentioned, Scrum and Scrumban ultimately require changes in the way both the business and the technology sides of the house approach things. Having an engaged player on the business side can be invaluable for overcoming challenges and evangelizing success across the organization.

  Scrumban modifies the weight of this factor substantially. To function properly, Scrum requires active engagement on the business side—its prioritization and feedback functions depend on it. Scrumban won't function well without business engagement, but it does afford teams the ability to create positive changes by allowing knowledge discovery to "stand in" for disengaged business players.

  In a similar vein, Scrumban is ultimately designed to respond to business demands. If the business isn't demanding change, then Scrumban's pragmatic "response" is to reflect the absence of any need to change (although a culture of continuous improvement will continue to spawn incremental changes to make what you're already doing well even better).

Initial Considerations

Many topics can be addressed during a kickstart. What constitutes "information overload" for a particular group will vary from context to context, and must always be judged against the level of ongoing support available following the kickstart (for example, you will likely cover more material if teams will have coaching support following the kickstart than if they will have none at all).

With these considerations in mind, incorporating themes that reinforce the service-oriented philosophy and the organization's core values will go a long way toward positioning teams to evolve more effectively. There are many ways to do this, and it's one area where an experienced practitioner or coach will really add value to the process.[3]

One special consideration for existing Scrum teams may be to introduce the concept of Sprint 0 (if this tactic is employed within the involved team or organization). There's much debate about the "appropriateness" of Sprint 0 in Scrum circles. These special sprints are often described as necessary vehicles for managing what needs to be done before a Scrum project can start. For example, the team may need to be assembled, hardware acquired and set up, and initial product backlogs developed. Purists' feathers may be ruffled by the notion of differentiating sprints by type and, more importantly, by the idea that any sprint would be structured in a way that fails to deliver value.

The Scrumban framework obviates the need for debate on these issues. The tasks associated with ramping up a new project can be easily accommodated and managed within the Scrumban framework as a special work type. If it's important for your environment to ensure Scrum ceremonies hold true to their Agile objectives, then Scrumban provides a ready solution.

It also makes sense to assess existing Scrum practices with an eye toward understanding their impact on performance. Larry Maccherone and his former colleagues at Rally Software have analyzed data from thousands of Scrum teams in an effort to assess how differences in the way Scrum is practiced affect various elements of performance. This data mining exposed some interesting realities about how our choices involving Scrum practices can influence various facets of performance. Incidentally, Maccherone's analysis is consistent with data mined from hundreds of teams using my own Scrum project management platform (ScrumDo.com).

Maccherone elected to adopt a "Software Development Performance Index" as a mode of measuring the impact of specific practices. The index is composed of measurements for the following aspects:

- ***Productivity*:** Average throughput—the number of stories completed in a given period of time
- ***Predictability*:** The stability of throughput over time—how much throughput values varied from the average over time for a given team
- ***Responsiveness*:** The average amount of time work on each user story is "in process"
- ***Quality*:** Measured as defect density

3. As previously noted, the Siemens Health Group case study (see the Appendix) represents a great example of how expert assistance played a critical role in contributing to the depth of the company's success.

How You Choose to Organize Makes a Difference

Especially For:
Managers & Executives

As noted, Larry Maccherone's analysis reflects a definite correlation between certain choices and software development performance. Although correlation does not necessarily mean causation, we can use these relationships as a guide for tailoring a kickstart process to address specific factors relevant to a particular implementation. Teams can be guided to position their visualizations and practices to better understand and discover which choices regarding Scrum practices are likely to work best for their desired outcomes.[4]

Determining Team Size

Humans are the slowest-moving parts in any complex organization. Teams can help us counteract this reality by making us smarter and faster. However, this outcome is possible only if we get teams right. Team dynamics is Scrum's strong suit—and an arena Kanban addresses only tangentially, if at all. In fact, providing a framework that helps us improve team dynamics is a great example of how Scrum boosts Kanban's capabilities.

To this end, size stands out as the most significant predictor of team success. There's a right size for every team, and like so many other aspects of managing systems, that size should be dictated by overall context. Even so, having guidelines doesn't hurt.

The military is a great starting point for gaining perspective on ideal team size. The basic unit in the U.S. Army's Special Forces is the 12-person team. The army arrived at this number after recognizing there are certain dynamics that arise only in small teams. For example, when a team is made up of 12 or fewer people, its members are more likely to care about one another. They're more likely to share information, and far more likely to come to each other's assistance. If the mission is important enough, they'll even sacrifice themselves for the good of the team. This happens in business, too.

The founders of Scrum understood these dynamics. Ask anyone in the Scrum community about ideal team size, and you'll hear 7 members plus or minus 2.

The reasons for maintaining small team sizes are valid, but not every 12-person military unit is right for a particular assignment. Likewise, not every 9-person Scrum team is right for a given environment. Your system ultimately will dictate your needs, and it may very well require expanding your teams to incorporate a larger number of specialists or address some other need.

---

4. In sharing this data, Maccherone has been diligent in emphasizing that there is no such thing as "best practices" relevant to all contexts. Each team and organization must discover what works best at any given point in time.

Scrumban supports this flexibility through its enhanced information sharing and visualization capabilities. Even the cadence of daily stand-ups is different: We've seen daily stand-ups in a Scrumban environment involving almost 100 individuals that are both effective in addressing organizational needs and capable of being completed within 15 minutes. This would be impossible in a Scrum context.

Iteration Length

Many organizations seek to establish uniform iteration lengths because they mistakenly believe this is a good approach for aligning different systems to the same cadence. Yet for many years, the general consensus among Scrum practitioners has been to recommend two-week iterations. What does the data tell us?

Rally Software's data shows that almost 60% of the software development teams using its tool adopt two-week sprints, and I've seen similar patterns among users of ScrumDo. The data also suggests a team's overall performance tends to be greatest at this cadence (Figure 5.2).

Dig more deeply, however, and differences begin to show up among the various components of the index. Throughput, for example, tends to be greatest among teams adopting one-week iterations (Figure 5.3).

Quality, in contrast, trends highest among teams adopting four-week cadences (Figure 5.4).

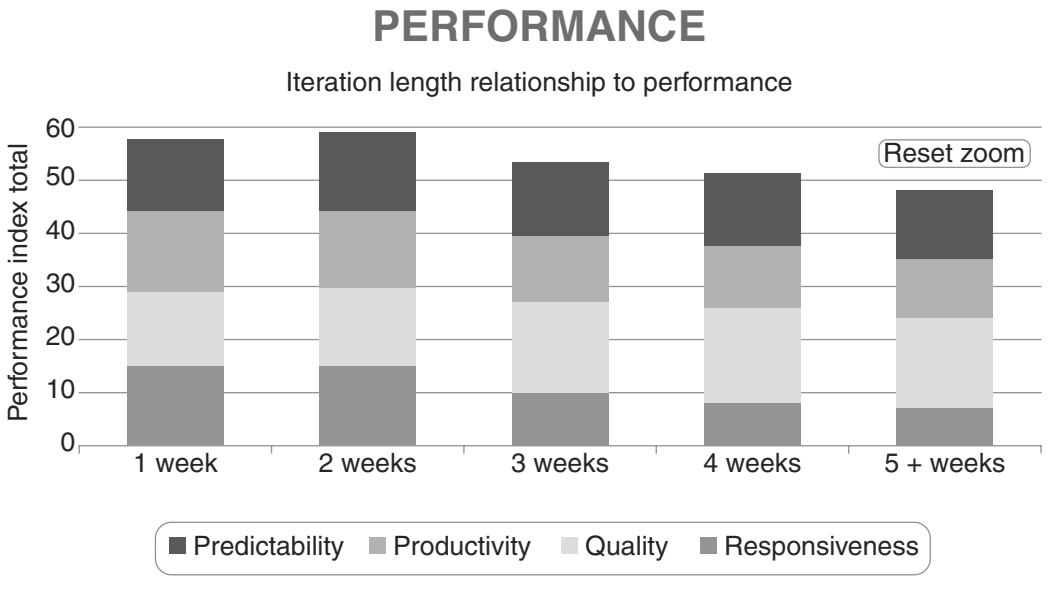


FIGURE 5.2 A view into the relationship between iteration length and team performance.

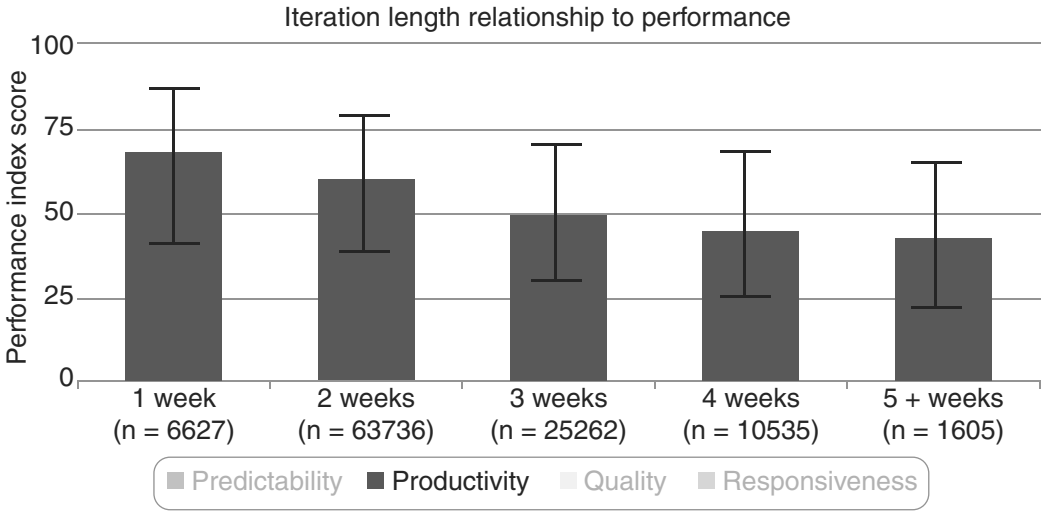


FIGURE 5.3 A closer look at productivity.

Source: © 2014 Rally Software Development Corp. All rights reserved. Used with permission.

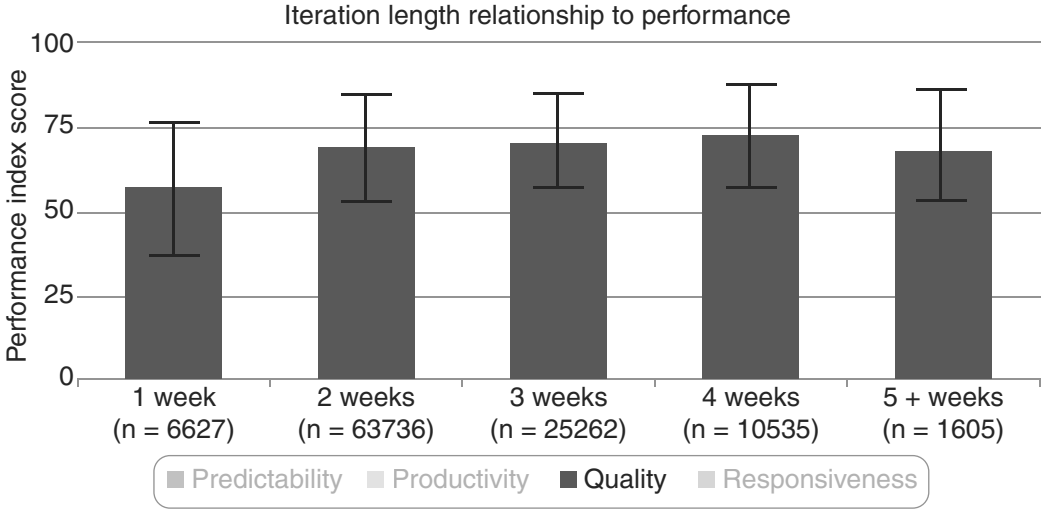


FIGURE 5.4 A closer look at quality.

Source: © 2014 Rally Software Development Corp. All rights reserved. Used with permission.

Every organization possesses characteristics that drive different outcomes, which is where Scrumban's visualization and added metrics can help teams discover what's best for their unique circumstances. The higher throughput typically associated with two-week cadences won't magically materialize if the historical lead time for most of your completed work items is three weeks. Similarly, delivering completed work every two weeks won't magically produce better outcomes if the customers can't accept it at that rate.

Also, be wary of inferences drawn from data associated with teams that have adopted longer iterations. For example, Frank Vega recently made an interesting observation on this topic, recounting instances where teams chose longer iteration periods as a means of opposing (consciously or subconsciously) Agile transformation efforts because agreeing to a shorter iteration cycle would counteract their position that nothing of any real value could be delivered in such a short time period.

Estimating Story Points

Scrum prescribes the Sprint Planning ceremony as an event that determines which specific stories can be delivered in the upcoming sprint and how that work will be achieved.

Although Scrum itself doesn't prescribe a particular approach or method that teams should use to assist with this decision making (other than expecting it to be based on experience), most Scrum teams use story points and team estimation techniques as components of this process.

Another process that teams use to aid their estimation efforts is Planning Poker—a consensus-based technique developed by Mike Cohn. With this approach, which is used primarily to forecast effort or relative size, team members offer estimates by placing numbered cards face-down on the table (rather than speaking them aloud). Once everyone has "played," the cards are revealed and estimates discussed. This technique helps a group avoid the cognitive bias associated with anchoring, where the first suggestion sets a precedent for subsequent estimates.

Scrumban enables teams to begin measuring the correlation between story point estimates and the actual time spent completing development, measured from the time work on a story begins until the story is completed. Some teams reflect a strong correlation between their estimates and actual work time. Others are more sporadic, especially as the estimated "size" of stories grows. Teams using exponentially based story size schemes tend to be more precise with their estimates. Comparing the two charts in Figure 5.5 and Figure 5.6, it's fairly evident that exponential estimation reflects a tighter band of variability in lead time—especially among smaller stories. The spectrum of variability is much wider across the Fibonacci series.

As with the data on sprint length, the way teams estimate the size and effort of work tends to correlate with overall performance (Figure 5.7).

It is not uncommon to find a lack of correlation between a team's velocity, the average number of story points completed during a sprint, and actual throughput. Nonetheless, there are many instances when the estimating process works well for

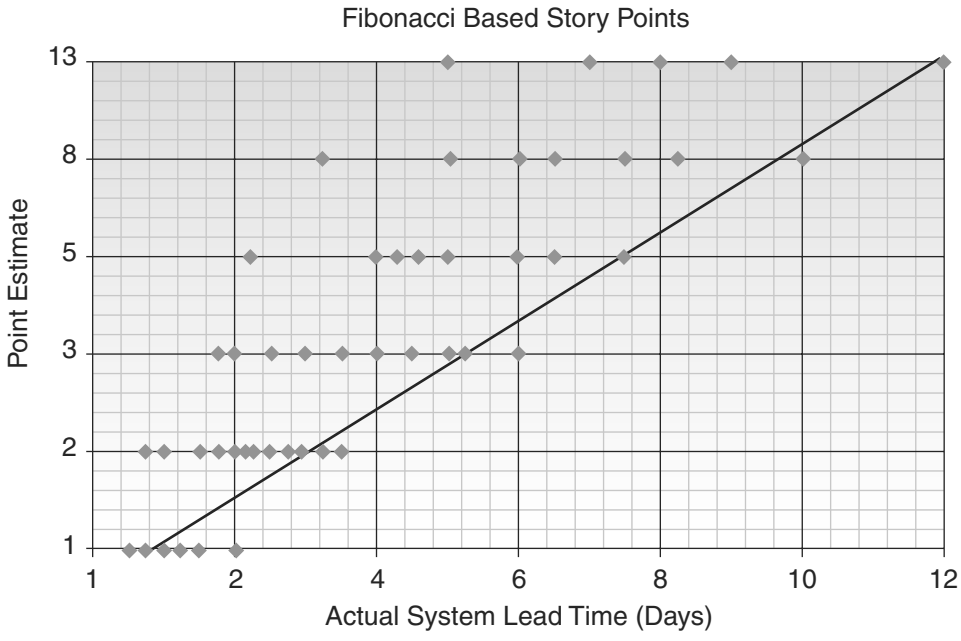


FIGURE 5.5 There tends to be only a loose correlation (if any at all) between estimated story points and actual lead time.

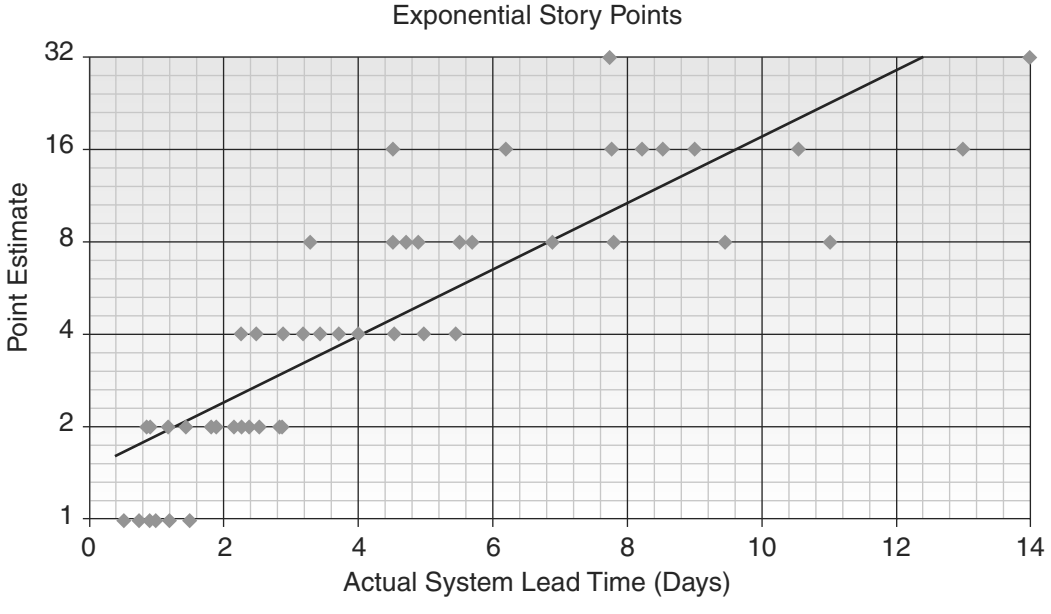


FIGURE 5.6 The correlation is slightly better in an exponential points scheme.

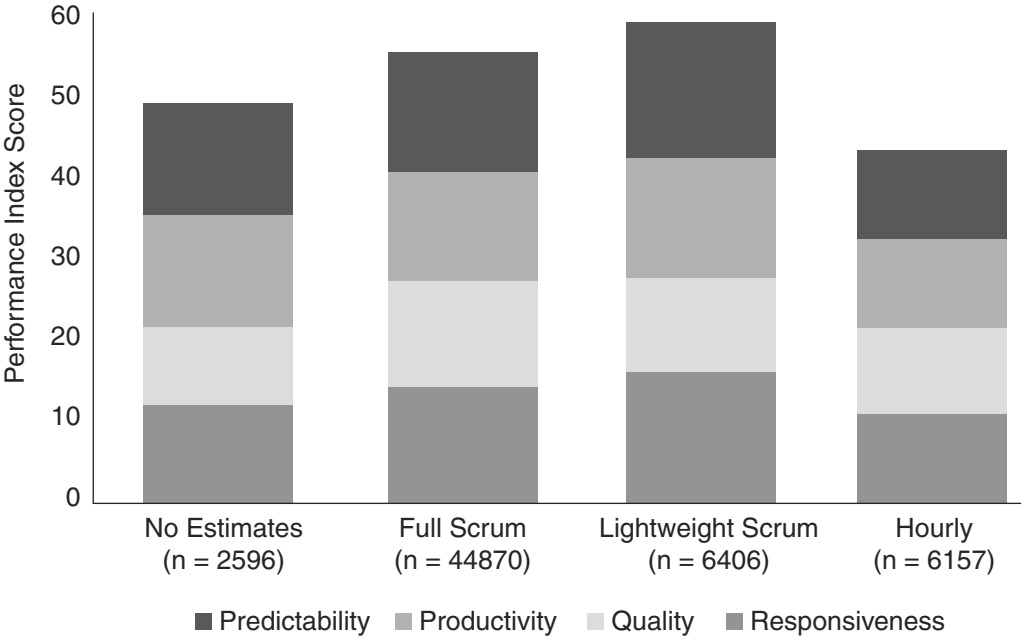


FIGURE 5.7 The relationship between team performance and estimation schemes.

individual teams. As mentioned previously, it's more challenging to work with this metric on a portfolio level involving multiple teams. Calculating relative correlation is one way that Scrumban helps teams gain a better sense of whether the ceremonies they engage in, such as estimation events, are providing sufficient value to justify the investment of time and effort.

Scrumban Stories: Objective Solutions

Objective Solutions' teams showed a lack of correlation between their story point estimates and actual lead time, but this discrepancy wasn't noticed until after the company's implementation of the Scrumban framework. As the organization opted to continue using team estimates for forecasting purposes, its solution for improving predictability was to apply a "velocity factor" derived from the difference between these values. To avoid the influence of Parkinson's law (the tendency for the time needed to complete work to expand to the time allowed for it), these velocity factors were not communicated to the team members, and only used by the product owners and Scrum masters engaged in planning.

Read the full story of Objective Solutions in the Appendix.

Contextual Responsibilities

As teams begin to employ Scrumban, members will assume a range of new "responsibilities."

First and foremost, team members must be groomed to challenge and question their team's policies, facilitating team interest in and ownership of how they work. Fortunately, this basic concept should not be foreign to people already used to a Scrum context. A great way to promote and support this mindset is through the establishment of "working agreements."[5]

Not surprisingly, Scrum's existing ceremonies and artifacts implicitly support the notion of working agreements. Scrumban's framework goes one step further, calling for the team to make such agreements explicit and to visualize them on their boards as appropriate. In many respects, they are akin to establishing an internal service level agreement between team members.

Ultimately, we want to develop leaders who help all team members acquire the ability to see and understand concepts like waste, blockers, and bottlenecks. Good leaders also ensure teams don't get stuck in a comfort zone, by prodding team members to always seek out ways to improve.

As previously mentioned, although teams will experience work improvements simply from employing Scrumban independently, an active management layer is essential to realizing benefits at scale. Systemic improvements are less likely to occur without a manager who maintains contact, helps resolve issues outside the team's domain, and oversees the extent to which teams devote time and effort on pursuing discovered opportunities for improvement. Setting expectations for this need at the kickstart helps ensure a more sustainable implementation.

Scrumban Stories: Objective Solutions

Like Siemens Health Services, Objective Solutions had successfully adopted Scrum/XP practices in its development operations but was struggling with a number of challenges at scale. The company adopted elements of Scrumban to address these challenges. Not surprisingly, this journey allowed the organization to improve many of its Scrum and pair programming practices that had been negatively influencing throughput and quality.

For example, Objective Solutions was experiencing challenges with managing the pair programming process: Keyboard work was not balanced, the time required to complete work unnecessarily expanded, and mentoring benefits were not being fully realized. Scrumban allowed this company to recognize specific problems, and to implement processes for managing its pair programming practices in the course of managing its regular workflow.

Read the full story of Objective Solutions in the Appendix.

5. See, for example, http://tiny.cc/AgileTeamAgreements.

The Kickstart Event

> **Coaching Tip!**
>
> This overview was prepared for initiative leaders, and covers broad, general guidelines. Adjust to your context as warranted.

As with any initiative, adoption of Scrumban should start with some type of "formal" event. Though simpler to launch than a direct transformation to Scrum, some elements of "education" and setting expectations are still necessary to ensure a smooth start. All team members should be physically present for any kick-off meeting. This is obviously more of a challenge with distributed teams, so some consideration must be given to communications and visualization. The topics that the agenda for a good kick-off event should, at a minimum, address are covered in the following sections.

Introductory Remarks

The kick-off event is an opportunity to set the stage for what follows. Remarks should reinforce at least two key concepts.

First, you want to underscore that this effort is solely about introducing and employing Scrumban in the context of your current ways of working, and represents little to no change in existing work roles or responsibilities. It's important to address this point up front to minimize the potential barriers to effective learning caused by fear or resistance to change.

Second, you should reinforce the fact that this event has just one achievable outcome—each team walking away with a clear visualization of how it works. As with threatened change, epic objectives create unnecessary psychological barriers. Communicating an achievable outcome up front will help engage participants early on and make the kickstart process considerably more meaningful. We define our "definition of done" for this step as follows:

- Each team member has a clear understanding of his or her team's purpose.
- Each team has created a visual representation of its ongoing work and current workflow.
- All team members agree on their most important and relevant work policies.

Current Concerns

Service orientation lies at the core of the Scrumban framework, so it's critical for teams to begin with a clear identification of their stakeholders and any dissatisfaction

they might have. Getting stakeholders to work with you, instead of against you, is one of your greatest tools for achieving continuous improvement.

Though adjustments should always be made based on the context of your environment, Klaus Leopold recently articulated one approach for visualizing stakeholder relationships that seems particularly informative and useful for introducing Scrumban to an organization.[6] Among the suggestions he offers are the following:

- Visualizing the power and influence of each stakeholder using different size cards.
- Visualizing the degree to which stakeholders support your initiative through a spatial reference point: The closer to the point they're positioned, the stronger their level of support. Although Leopold's article specifically speaks to stakeholder relationships relative to undertaking a change initiative in general, there's no reason this approach can't be undertaken for any initiative.
- Visualizing the frequency of the relationships among stakeholders with different style lines (e.g., dotted for infrequent or tenuous connections, other styles for different grades).
- Visualizing the quality of the relationship with special symbols (e.g., friendly or adversarial, healthy and strong or tenuous and weak).

Why engage in this evaluation of stakeholders? Because you need to understand your stakeholders' sources of dissatisfaction, and find the most effective ways to work with them toward resolving those issues. In fact, teams should change their work policies and work visualizations only if doing so will help resolve areas of dissatisfaction.

Visualizing the dynamics associated with key relationships also creates a clear view of who holds the most power, the current state of each relationship, and the steps that must be taken to move closer to a solution. It helps prioritize an approach to resolving dissatisfaction.

Next-Level Learning:

Evolving to More Advanced Practices

If you're looking to apply an even greater level of calibration to the process, this arena is ripe for applying a model like the Cynefin framework to provide information on how best to manage a given relationship.

In most circumstances, it is appropriate to facilitate discussions around current issues. Simply invite members to identify the top three to five issues that represent irritants or impediments to their work. Any issues that relate to how work is done or how the team interacts with other groups can be set aside for possible discussion, as

6. See http://tiny.cc/StakeholderDissatisfact.

these represent way-of-working or policy matters the team will be addressing during the kickstart. Any others should be tabled, as they don't relate to issues relevant to the workshop.

Defining Purpose and Success Criteria

Teams and organizations that lack a common purpose typically cannot achieve or sustain high levels of performance. Any exercise that moves them toward a common understanding of purpose and criteria for success is all that matters. I like to have teams answer the question, "Why does our team exist?"

You should skip this step only when you're sure everyone agrees they're on a common journey to pursue incremental change toward working more effectively.

Identifying How Work Is Done

The goal for this segment is to have teams gain a realistic understanding of their current situation—in other words, how they actually perform work versus how they *think* they perform work. We get to this point by introducing a systems perspective—by asking the team to visualize itself as a closed system having several basic components (Figure 5.8). This exercise may be the first time many teams acquire a realistic sense

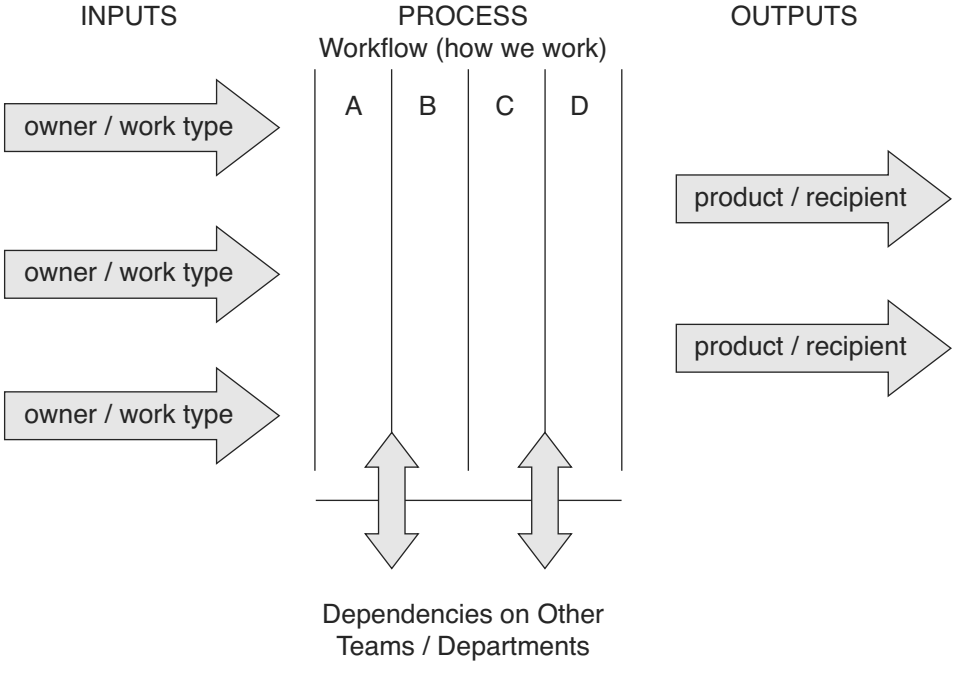


FIGURE 5.8 Visualizing a closed system.

of the complexity of how work flows in and out of their domain, plus the full scope of what they're responsible for producing on an ongoing basis.

The team's articulation of upstream demand represents which work they're asked to perform, who makes those demands (internal customers, external customers, or a combination of both), how those demands are communicated, and what their relative frequency and quantity are. These understandings are used to categorize work types later in the workshop.

Defining the downstream end reinforces the notion of recipients as consumers or partners in a process, thereby reinforcing Kanban's concept of knowledge work as service delivery. These initial definitions of system boundaries clarify the scope of the team's responsibility—where their work begins and ends.

Articulating how the team performs its work should command most of the team's attention. Teams should be guided to creating a simple, high-level visualization of the stages that each type of work must pass through before completion; this will be the foundation upon which their working board is built. Simple and abstract is the key—no more than 2–3 different kinds of workflows and no more than 7–10 stages in each.

Focusing on Work Types

A good kickstart process will incorporate activities that ensure team members acquire a common understanding of work types and their significance. It may be relevant to your organizational context to have teams identify and manage their work using predefined types, but in any case it's important for the team to understand the nature of the demands placed upon them. Naturally, this is an ongoing process. Common modes of approaching this include assessing which work types meet the following criteria:

- Have the most value for their customers or partners
- Are demanded the most and the least (in terms of quantity)
- Are usually more urgent than others
- Are best aligned with the team's purpose (the kind of work the team should do to a greater extent)
- Are least aligned with the team's purpose (the kind of work the team should do to a lesser extent)

It may be relevant to your organizational context to have teams identify and manage their work using predefined types. Nevertheless, regardless of how you elect to proceed, it's ultimately most important that they understand the nature of the demands being placed upon them. Naturally, this is an ongoing process. Common categorizations include the following:

- By source (e.g., retail banking, product X, maintenance items)
- By size (e.g., in terms of effort)
- By outcome (e.g., production release, analysis report)

- By type of flow (e.g., development, maintenance, analysis)
- By risk profile (e.g., standard work, urgent work, regulatory compliance)
- By relevance (how closely the work is aligned with team purpose)

Whatever scheme you choose, categorization provides a frame of reference against which an appropriate balance of work in progress can be created and managed.

Basic Management

It's one thing to have teams begin visualizing how they work; it's another thing to provide them with a framework for using these visualizations to discover better ways of managing it. I recommend using the GetScrumban game (Figure 5.9) to introduce teams to the basic principles behind managing their workflow. (Full disclosure: I designed this game.) It's usually employed in tandem with other "classroom" exercises to illustrate and emphasize key principles and practices.

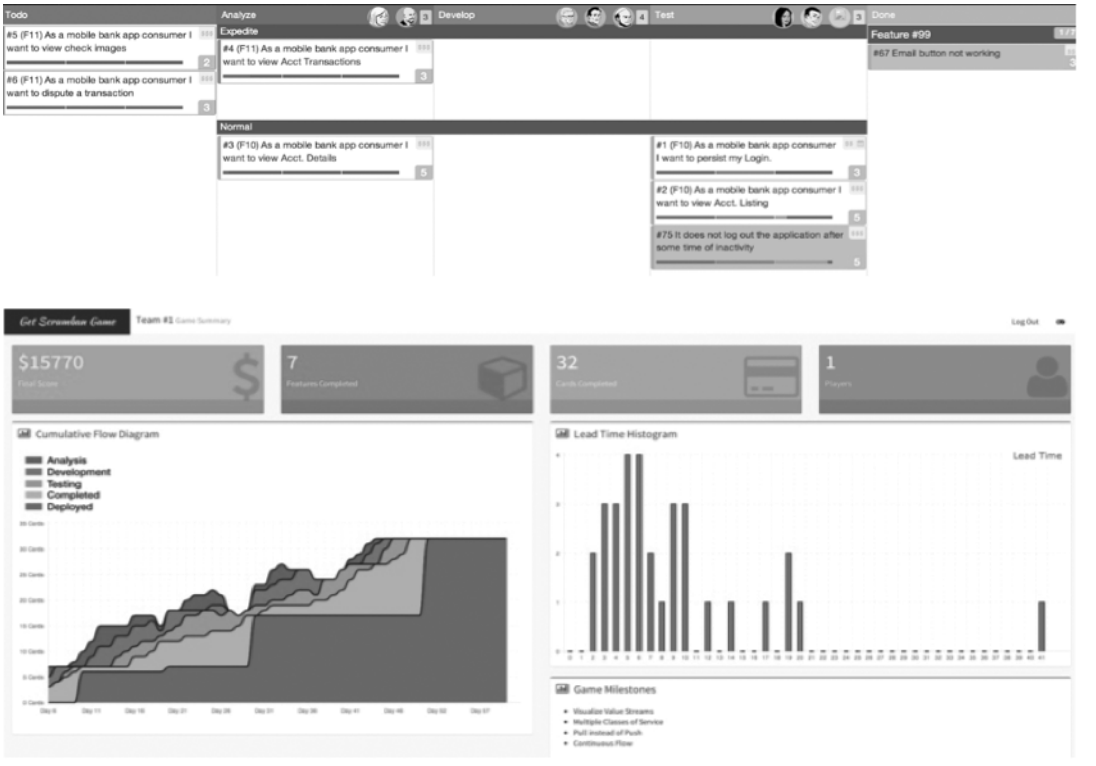


FIGURE 5.9 The GetScrumban game lets players experience the typical evolution of Scrum teams after introducing the Scrumban framework into their way of working.

Source: GetScrumban.com.

The GetScrumban game simulates how a software development team that employs Scrum as its chosen framework can use Scrumban's core principles and practices to amplify its current capabilities, overcome common challenges, or forge new paths to improved agility. The game allows players to experiment with and experience the impact of these principles and practices:

- Expanded visualizations
  - Value streams
  - Types of work
  - Risk profiles
- Pulling work versus assigning work
- Evolutionary adjustments versus radical change
- Cost of delay versus subjective prioritization
- Distinct classes of service versus a single workflow
- Continuous flow versus time-boxed iterations
- Value of options

Whether you use an interactive tool like GetScrumban or employ some other mode of instruction, your teams should walk away with an understanding of the following:

- **Concepts Already Familiar to Scrum Teams**
  - ***Work items/cards (user stories)*:** These are typically visualized on physical boards in the form of a sticky.
  - ***Work size estimate (story points)*:** As most Scrum teams already engage in some form of estimation, the concept of estimating (and the notion of breaking larger stories down into more manageable sizes) should be very familiar.
  - ***Definition of done*:** A short checklist of standards that must be present for a work item to move from one column to another.
  - ***Daily stand-ups*:** Scrumban stand-ups tend to eliminate declarations of status (what each team member completed, what the team member is committing to complete, and the identification of impediments) because all of that information is already visualized on the board. Instead, stand-ups evolve to focus on how well work is flowing and which actions the team can take to improve overall flow and delivery of value.
- **New Concepts**
  - ***Work type*:** Often represented by the color of a work card; reflects the mix of work in progress. Visualizing and actively managing the mix of work (i.e., standard user story, bug fix, maintenance item, estimations) is usually a novel concept for Scrum teams.

- ***Workflow***: Columns on a work board represent the value-adding stages work passes through to completion. These usually start with "Ready" and conclude with "Done," "Ready to Deploy," or some similar terminology. Scrum teams often welcome this change because story progress and individual contributions toward it are made more visible.
- ***Pull***: Though some Scrum teams may be familiar with pull-based systems, many others are new to this mechanism. Pull mechanisms avoid clogging the system with too much work. Rather than work being "pushed" onto a team by those with a demand, the team selects work to pull into their work stream when they have the capacity to handle it.
- ***Ready for pull***: These "holding" areas are visualized as columns within a column. Typically used when a handover occurs (such as from a development phase to a test phase), they help manage bottlenecks.
- ***Definition of ready***: A short checklist at the bottom of a "Ready" column that visualizes the relationship a team has with those requesting work. The definition should specify the information or resources a team needs to effectively begin working on an item. Though many Scrum teams may employ definitions of ready in their work, they are rarely explicitly defined and visualized within a working framework.
- ***Blockers***: Flags that indicate when work on an item is suspended because of dependencies on others. Blockers are usually visualized as an additional sticky or magnet (pink or red) on a work item. The purpose is to call attention to such items so the team attends to removing the impediment. Some Scrum teams may already visualize blockers on their task boards.
- ***Classes of service***: Different "swim lanes" used to call out different risk profiles associated with given work items. We can choose to visualize separate classes of service to reflect and manage risk better (e.g., helping to ensure higher risk profiles attract more attention from the team than lower risk profiles. Similarly, you may want to help the team recognize it's okay to take longer to complete lower-risk items.
- ***Explicit WIP limits***: Teams should limit the amount of work in progress at any given time. We can do this by establishing explicit WIP limits across the board, within each column (preferred), within each swim lane/class of service, by work type, by team member, or any combination. Limiting WIP improves flow efficiency (by reducing or eliminating the cost of context-switching, among other things).

Common Language (Optional)

If you're looking to align Scrumban practices across a large number of teams, it's usually beneficial to establish a "common language" around common concepts. It's possible to borrow some terms from Scrum (for example, teams might all carry a

"backlog" or items that are ready to be worked on). Similarly, it may make sense for teams working on different aspects of the same program to use the same visualization scheme and share common policies.

Visualization Policies

One of Scrumban's core practices is to ensure that all work policies are explicit. We do so to ensure that everyone is on the same page, and that the work policies can be easily remembered and shared with others. Common practices include the following:

- ***Work items*:** Most practices will be natural carry-overs from Scrum:
  - Due date (if any).
  - External reference (e.g., from a management/tracking tool).
  - Size of work (e.g., story points or person-hour estimates).
  - Start date (important to track for measurement purposes).
  - End date (date work was fully completed—some metrics can be impacted by how you measure this, but any agreed-upon policy is sufficient when starting out).
- ***Workflow*:** The basic elements should already be incorporated in the systems diagram the team developed earlier. Some columns may be adjusted and rows added, however, as the team's understanding develops.
  - Pull Criteria: Scrum practitioners familiar with Definition of Done can optionally break it up into more granular lightweight "Pull criteria" visualized as a combination of mandatory and optional conditions before which a pull can be made.
- ***What* not *to visualize*:** Cluttering up your visualization with unnecessary items defeats the objective of bringing greater clarity and understanding to how you work. Although teams should capture as much of their work as possible, there can be legitimate omissions, as in the following examples:
  - Administrative activities (such as meetings unrelated to ongoing work). However, there may be great value to capturing how much time meetings are taking away from actual work, or whether certain resources are more encumbered than others.
  - Short, ad hoc work (5- to 10-minute requests or incidents). As with administrative activities, there can be value in capturing these items. Measuring the number of such work items in the course of a day or week could reveal a significant and ongoing demand that would otherwise fly beneath the radar.

Frequency of Synchronization

Daily meetings are as much a ceremony with Scrumban as they are with Scrum. Unlike in Scrum, however, teams can move beyond sharing status and making

commitments to collaborating on impediments to workflow and recognizing opportunities for improvement. This requires the team's visual board to be in sync with reality. Some development teams may be able to get by with once-a-day synchronization, whereas others will need real-time updates. Decisions made in this context can impact tool choices and other related factors.

Create a Working Board

The kickstart session is an ideal time to assist teams with setting up their working boards. The sooner a team starts to see and work with actual items, the more relevant the process becomes. This effort typically involves the following steps:

- ***Drawing the workflow***: Creating columns that represent the value stream of the team's work process.
- ***Creating the board***: I recommend that the teams create the board together, whether it's an electronic tool or physical board. This accelerates team learning and ownership.
- ***Ticket design***: The information to incorporate on a work ticket will vary from team to team. We discuss these considerations in more detail in Chapter 7.
- ***Adding current work*** (self-explanatory).

This is also an ideal time for teams to establish their definition of ready and definition of done for work items and lanes. Ready definitions can be easy to neglect, but they help avoid potential blockages once work begins. Teams might also discuss prioritization, but in a Scrum context this issue should have been already addressed by product owners.

Way of Working Policies

It's critical that all team members agree how work will be handled in their visual boards so that it is done in a consistent manner. Areas to address include the following:

- Which individual(s) will be responsible for managing the ready buffer (placing new work items in the buffer and prioritizing them for the team to address as capacity allows). This topic is not relevant for teams that continue to use time-boxed sprints.
- When and how the ready buffer will be replenished. For teams that continue practicing Scrum, this usually coincides with the sprint planning process. In complex environments, developing policies could become quite involved. The immediate objective is simply to have a workable starting point.
- Which individual(s) will be responsible for managing completed work (especially important when work is to be forwarded to downstream partners).

- How ad hoc work or requests from outside normal channels should be managed. Consideration must be given to the needs of the system making the request.
- When work should be pulled from the ready buffer (typically whenever a team member has capacity and can't contribute to any ongoing work). Consideration should be included for managing WIP limits and what should occur if exceptions are to be made.

Most of these considerations involve assessing risk (addressed further in Chapter 6). Having teams explicitly address risk as part of the kickstart process is the best way of ensuring more considered management as they mature. At a minimum, teams should be encouraged to explicitly articulate how work will be prioritized and which considerations are expected to be addressed as part of this process (i.e., is prioritization based exclusively on market/business risks, or should the decision incorporate an assessment of risks associated with the underlying technology, the complexity of the work, the team's familiarity with the domain, and other factors).

Limiting WIP

Coaching Tip!

This is an especially important topic.

Setting explicit limits on work in progress will be a new concept for Scrum teams, and the science behind this mandate can be counterintuitive. For teams already practicing Scrum, it may be beneficial to point out how the Sprint ceremony is an implicit WIP-limiting mechanic. The idea is not to dive into detail, but rather to provide enough of an overview so the team understands why limiting work in progress matters.

"Stop starting and start finishing" should become every team's mantra. Pull mechanisms are one way to ensure new work items are started only when a team member has capacity to do the work, thereby enabling the team to eventually attain a stable WIP level that matches its total capacity. Explicit WIP limits are another strategy (but should really be reserved for more mature teams, as they require a more complete understanding and practice of many core concepts).

To help ensure system agility, it's essential to maximize your options. The more tightly work is packed within a system, the less agile you become (tightly packed work reduces your available options to respond to changes in circumstances). Limiting WIP is one mechanism for maintaining a sufficient amount of slack to ensure a smoother flow through the system.

A commonly used approach when introducing teams to the concept of explicit WIP limits is to establish them based on available resources or some other factor

that's not associated with actual demand and capacity. As with most things, the best approach will be dictated by the team's specific context.

As the team performs work, circumstances may call for violating WIP limits. These are learning opportunities, and should always trigger a discussion. Perhaps current limits are too low and should be raised. Or perhaps circumstances are such that the need constitutes a one-time exception. Regardless of the context, WIP limits represent an essential constraint that forces teams to improve their process and way of working.

Practice Tip

The amount of work a system should have in progress at any given time is a matter of balance. Just as tightly packed work can impede flow, so carrying too many options can represent waste. The proper balance is achieved over time, and is not something to target when starting off. Nonetheless, understanding what proper balance is expected to resemble is important to establishing any "proto" constraints with which you elect to begin.

Teams will confront some common challenges with regard to establishing and abiding by WIP limits (we address these in greater detail in Chapter 7). These include the following issues:

- ***Variability***: In the form of demand, the work, the risks, and numerous other factors.
- ***Constraints***: Constraints ultimately determine system capacity. In knowledge work, they tend to move around, making them difficult to identify and manage. This makes discovering and establishing the right WIP limits very challenging.
- ***Human nature***: People are funny. They often mask their true desires and intentions in less than obvious ways.

Planning and Feedback Loops

Scrum practitioners are already used to daily check-ins. Employing Scrumban, however, presents the opportunity to shift the meeting's focus from status and commitment to more proactive planning. The kanban board already provides status information and should detail who is working on what. Impediments should also be visualized. Consequently, the focus of the team's discussion can shift to a collaborative effort geared toward identifying potential impediments, dealing with requested exceptions to policies, and otherwise addressing discoveries about how work is being performed.

If your Scrum teams are made up of inexperienced practitioners, it's possible they don't know whether their existing stand-ups are even effective. If the teams to which

you're about to introduce Scrumban fall into this category, the kickstart is an opportunity to help them improve.

Jason Yip, a principal consultant at the firm Thoughtworks, has effectively summarized patterns to establish for a good stand-up (easily remembered using the mnemonic GIFTS):

- ***Good start***: Good stand-ups should be energizing, not demotivating.
- ***Improvement***: The primary purpose of the meeting is to support improvement, not to discuss status.[7]
- ***Focus***: Stay focused on the right things, which should be to move work through the system (rather than dwelling on pointless activities).
- ***Team***: Good stand-ups foster effective communication and collaboration. If people aren't helping one another during stand-ups, something is awry.
- ***Status***: Stand-ups should communicate a basic sense of what's going on. As previously noted, the actual conversations move away from this information in Scrumban (i.e., this information should be communicated through the kanban board).

Scrum's time-boxed Sprint remains the main vehicle for coordinating the delivery of completed work and replenishing the team with new work. Over time, teams can opt to modify the process for replenishment and commitment to delivery (and even de-couple these cadences) to adopt approaches more focused on actual demand and capacity.

Sprint Reviews and Retrospectives are existing feedback loops within the Scrum framework that we tweak in some measure when practicing Scrumban. For example, whereas Sprint Reviews are focused on soliciting customer feedback on the delivery of completed product, in Scrumban we incorporate the concept of reviewing overall service delivery (borrowing from the Kanban Method's Service Delivery and Risk Review cadences). Similarly, the Sprint Retrospective takes on more of the flavor of the Kanban Method's Operations Review.

It is also important to mention the importance of conducting organizational level feedback loops, such as the Strategy Review cadence integrated within the Kanban Method.

Individual Flow (Optional)

Some of the concepts and techniques Scrumban employs to give teams greater control over their collective focus and workflow are not obvious and are sometimes

7. To this end, helping team members learn how to effectively engage in difficult conversations can be immensely beneficial. See our references in the Appendix for additional guidance in this area.

counterintuitive. Devoting a portion of your kickstart program to techniques for managing personal workflow is one way to enhance appreciation of their effectiveness. Topics of particular relevance include the following:

- Managing energy and not time (introduced via the Pomodoro technique, for example)
- The power of habits in knowledge work (by way of a brief introduction to disciplined approaches in problem solving such as A3 Thinking, for example)

Wrapping Up

Kickstart workshops should always be closed with a summary of what the teams achieved and the next steps that will be taken moving forward. In all instances, some degree of continued coaching and guidance is necessary for teams to optimize their understanding and practice.

Some Final Thoughts

Coaching Tip!
Important Concept Ahead

We've encountered a common misconception among Agile consultants and coaches regarding Scrumban implementations—the notion that because kanban systems are grounded in queuing and systems theories, there is no value in implementing them until the systems in which they're used are stabilized.[8]

Key Take-Away

As Deming recognized decades ago, systems should be stabilized before undertaking efforts to improve them. Unfortunately, many Agile experts fail to recognize that kanban systems provide a path to stability in and of themselves (from which we can apply scientific principles in pursuit of more considered improvements).

For example, WIP limits should ultimately be established based on the characteristics of the stable system to which they apply. However, initial-WIP limits can be used to bring a system into stability. I've pursued this path in many contexts, and

8. This mindset was recently communicated by a leading consultant in the Boston Agile community during a regular monthly gathering.

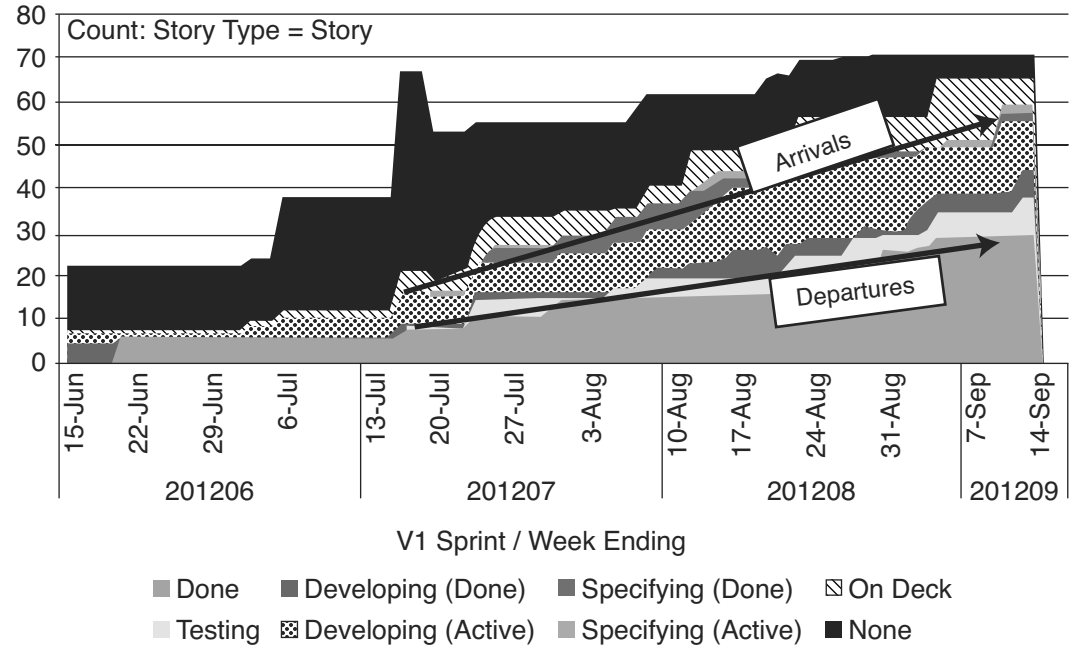


FIGURE 5.10 A cumulative flow diagram showing effort being committed to new work increasing faster than existing work can be completed.

this approach is specifically cited in the Siemens Health Services Scrumban story. It's worthy to call out some of the particulars here.

By way of background, the Siemens group elected to refrain from imposing WIP limits on their initial rollout. They soon discovered that the absence of WIP limits did nothing to stabilize or reverse their existing trend of ever-increasing cycle times. A cumulative flow diagram generated from their data showed the system was out of balance, with teams accepting new work faster than they were delivering completed work (Figure 5.10). Past patterns of increasing cycle times had not been influenced.

Upon establishing initial-WIP limits, the teams immediately began to see a stabilization in both system lead times[9] and overall system performance (Figure 5.11 and Figure 5.12).

The moral of the story is clear: Yes, systems need to be stable before we can improve them, but they also afford us mechanisms to achieve the stabilization needed to undertake our true journey.

9. Depicted here as cycle time. The terminology in the industry can be quite confusing. Cycle time here is the time taken to complete a work item. Refer to Chapter 7 for more precise definitions.

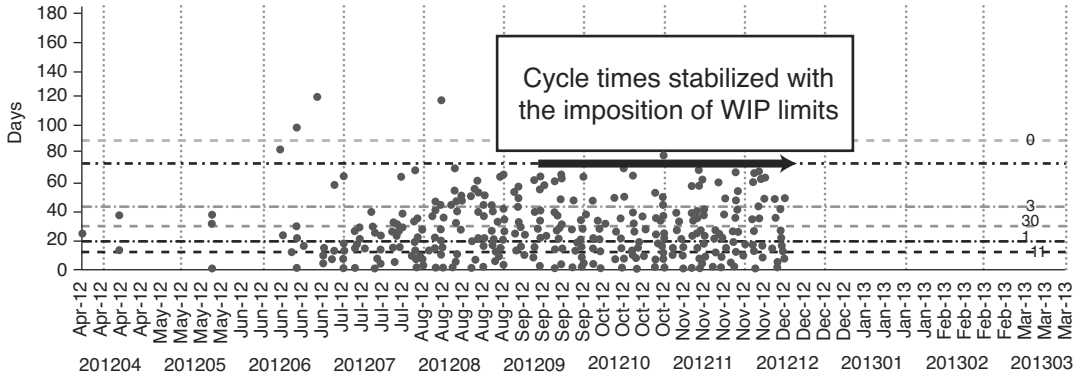


FIGURE 5.11 System lead times being stabilized.

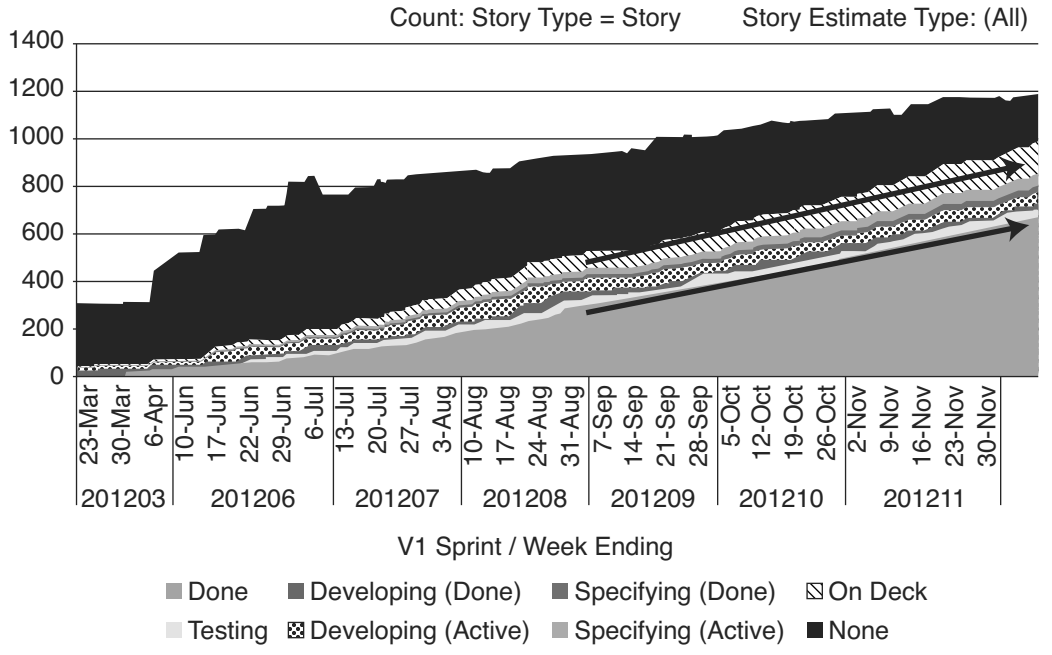


FIGURE 5.12 Cumulative flow diagram indicating a stable system.

Tying It All Together

This chapter outlines one approach for introducing Scrumban to a team or organization. Though not explicitly called out, it assumes the rollout is taking place in a mid-size or larger organization with an objective of catalyzing change to key outcomes across the enterprise.

Software engineers, project and product managers, and the other professionals with whom they interact are generally intelligent and capable people. While this chapter has provided enough detail for the uninitiated to begin introducing Scrumban on their own, don't discount the value of calling in outside experts, especially those who have seasoned systems thinking capabilities and a deep understanding of the Kanban Method. Indeed, the folks at Siemens Health Services made a point to call this out in the case study of their own experience.

This page intentionally left blank

INDEX

B

C

D

E

F

G

H

M

Q

R

S

T

U

V

W

Z