Extracted from:

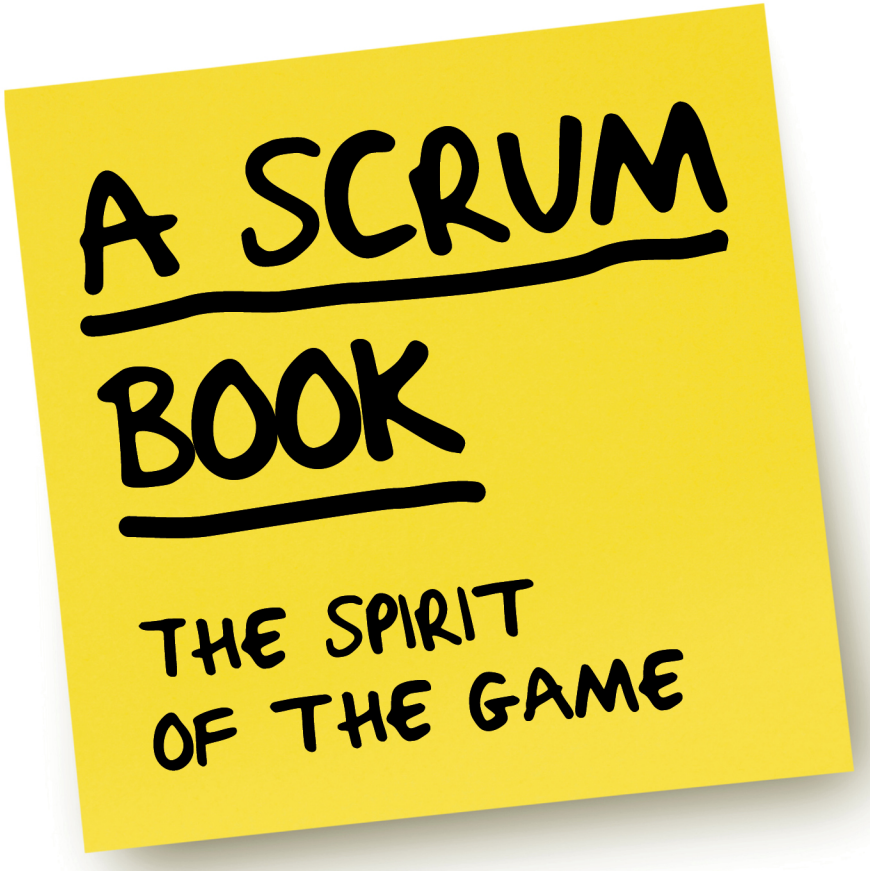
A Scrum Book

The Spirit of the Game

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers



Jeff Sutherland
James O. Coplien
The Scrum Patterns Group

edited by Adaobi Obi Tulton

A Scrum Book

The Spirit of the Game

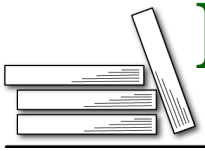
Jeff Sutherland
James O. Coplien

Lachlan Heasman
Mark den Hollander
Cesário Oliveira Ramos

and The Scrum Patterns Group:

Esther Vervloed, Neil Harrison, Kiro Harada, Joseph Yoder,
June Kim, Alan O'Callaghan, Mike Beedle, Gertrud Bjørnvig,
Dina Friis, Ville Reijonen, Gabrielle Benefield, Jens Østergaard,
Veli-Pekka Eloranta, Evan Leonard, and Ademar Aguiar

The Pragmatic Bookshelf
Raleigh, North Carolina



Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at *https://pragprog.com*.

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Managing Editor: Susan Conant
Development Editor: Adaobi Obi Tulton
Copy Editor: Sean Dennis
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-671-6
Book version: P1.0—August 2019

Introduction

This is a book about Scrum—a simple but powerful way for people to work together. Scrum is about helping small teams create, build, and evolve a product, one slice at a time. Scrum is as much about people as it is about the products they build and use. It defines a work environment where teams can challenge themselves to become better and better at their work over time. Its simplicity is much of its power: *The Scrum Guide*[1] is a succinct definition of Scrum fundamentals and it's only 19 pages long.

Scrum is not a development method, but rather a product development framework. A method prescribes steps that transform a problem definition into a solution. Scrum, on the other hand, is a framework for the team to continuously evaluate its product and evolve it into a *better* solution. More than that, Scrum also guides the team to continuously evaluate *how* it builds the product, and to creatively adapt its process to work more efficiently, deliver ever more responsively, and to grow product quality. Scrum defines how a self-managing team might best structure its work to deliver product increments to create the highest possible value for all stakeholders. Scrum focuses outward on the product that will add value to the community and to the end user, while also reflecting on how to make the development process ever more powerful, and the workplace a fulfilling and happy place. Scrum shines in complex domains where product requirements are unclear. It dissolves this uncertainty with frequent feedback, short iterations, time-boxing, and continuous workflow. Scrum is certainly the most widely adopted of all agile approaches: journalism, advertising, software development, and automobile construction are all examples of industries and professions where Scrum has made inroads.

1. Jeff Sutherland and Ken Schwaber. *The Scrum Guide*. ScrumGuides.org, http://www.scrumguides.org (accessed 5 January 2017).

Scrum—Through the Lens of Experience

Scrum has its roots in Japanese manufacturing. It takes its foundations broadly from the Toyota Product Development System, and in particular, from the Toyota Production System. Scrum adoption has grown rapidly since its inception in 1993 and its introduction to the public in 1995. Yet, it is still new to many organizations. For those new to Scrum, it may be a mystery to identify where to start. For organizations already using Scrum, it can be a challenge to find out where to focus improvement. Over the time that we, the book's authors, have been using Scrum (and we have been using Scrum since its inception), we have come to understand the networks of patterns that make it work: *how* Scrum works, *why* it works, and *how* best to apply it in daily practice.

We have several person-centuries of combined experience with Scrum. This book gathers that collective experience into proven solutions called *patterns* that we have distilled from observing many Scrum Teams—both their successes and failures. These solutions will help you implement and improve your use of Scrum, whether you are a beginner or an experienced practitioner. Though pragmatic and grounded in experience, this book also stands on Scrum's deepest foundations and reflects contributions from many early shapers of Scrum, including its inventor. As a rationalized oracle, this book hopes to help dispel many of the widespread myths about Scrum and its practices.

The solutions in this book not only draw on prior art and publication but also tap into the vast experience of a broad international community of product developers. The patterns come from our work with Scrum Teams worldwide, from our home bases in Japan, the Nordics, the U.K., Portugal, Canada, U.S., Netherlands, and Australia—and from our experiences on every continent except Antarctica. We have observed these patterns in many contexts, from organizations with thousands of staff members to small teams of three people; and in dozens of industries: telecommunications, banking, education, machine automation, and countless others. Each pattern has been through a process of detailed review by up to a dozen people, each of whom applied their collective dirty-hands experience with Scrum to relentlessly refine each one. Dozens of candidate patterns didn't make it to the book, as they rose only to the level of anecdotal experience, or lacked empirical validation, or were just precursors to greater patterns. We believe that the patterns in this book have something special. The pattern community calls it "the Quality without a name," or sometimes just "the Quality," a kind of Wholeness that aspires to day-to-day excellence.

Adaptive Problems, Adaptive Solutions

The Scrum Guide defines Scrum as "a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value."[2] This definition describes both Scrum's assumptions and the perspectives we develop from using it. One way to recognize a complex problem is to recognize it as one that you don't fully understand until seeing its solution, and that you can't derive the solution from first principles ahead of time. Solving the problem begs exploration and feedback. The problem's dynamic element demands adaptive solutions that change over time and with our engagement. Our current solution meets our current needs—but these will change. An example of this kind of problem is new product development: it may be easy to see why the product was a success after it is successful, but to be certain beforehand is impossible.

Beyond the Rulebook

Speaking of *The Scrum Guide*, we view it just as the rulebook for Scrum. It's important to understand the rules, and it's even useful to follow them most of the time. But reading the rulebook of chess won't make you a great chess player. After learning the rules, the player then learns about common strategies for the game; the player may also learn basic techniques at this level. Next is learning how to combine strategies you learn from others while maybe adding some of your own. Ultimately, one can transcend any formalism and proceed from the cues one receives from one's center, from one's instinct. This book is for those who want to understand the rules more clearly; for those who want to grow by combining ideas and by evolving the fundamental structure of the system, rather than by merely refining technique. Jeff Sutherland points this out:

> In recent years the Scrum Patterns Group has evolved a comprehensive set of patterns for Scrum that allow teams to try proven approaches that have worked in many companies. While *The Scrum Guide* describes the basic rules of Scrum, the patterns amplify the guide by showing teams how to solve problems in a specific context.

So *The Scrum Guide* is the rulebook, and the patterns look beyond the rules. These patterns shape the on-the-job-training of making Scrum work for you on your way up the ladder of ongoing improvement. Some day, long from now, you may even outgrow these patterns as you evolve them and define your own. There are no points for doing Scrum, and these patterns are the gate through which

2. Jeff Sutherland and Ken Schwaber. *The Scrum Guide*. ScrumGuides.org, http://www.scrumguides.org (accessed 5 January 2017).

a highly driven team passes on the road to the top echelons of performance. Patterns do their job when they help people honestly and critically search within themselves on their path to excellence. One ignores the patterns at one's peril, but only an unthinking organization follows them slavishly.

Patterns of Scrum

We describe elements of the Scrum framework using a form called *patterns*. What is a pattern? One simple definition is that a pattern is a repeatably applicable solution to a problem that arises in a specific context. Christopher Alexander, a building architect, popularized the modern notion of patterns as a way for communities to celebrate, socialize, and refine the design norms of their built world, in *The Timeless Way of Building [Ale79]*. The pattern-writing form (see Pattern Form on page xii later in this chapter) structures the written insights that describe how people build things—things that aspire to a quality of Wholeness which pattern folks call "the Quality without a name." You might have experienced the feeling of this Quality if you have seen some result in an area in which you have deep experience, that just "feels right" or gives you a deep sense of warmth, contentment, and confidence.

Alexander's work focused on constructing towns, neighborhoods, and buildings, but the underlying theory extends to anything built in a community. Beyond that, patterns structure broad solutions to recurring general problems, while each one unveils a concrete, proven solution in the reader's mind. Patterns in general describe forms and how designers compose them as they create and evolve structure in their worlds. This book focuses on the two main structures that organizations build to realize Scrum: the organization with its roles, relationships, and periodic gatherings of people, and the Value Stream that choreographs these gatherings over time and connects them to their wellsprings and to their effluent out to the market. Scrum patterns are the forms we create and compose when we structure both the development organization and the development process.

Each pattern helps you focus your thinking on a single, well-contextualized problem at a time. Working in small steps reduces risk and helps teams move forward with confidence in "process improvement." Each pattern describes the solution in enough detail to bring you to the point of that "aha" where you recognize the solution deep within yourself. Beyond that, the pattern suggests how it might change the organization, and it looks ahead to both negative and positive consequences of doing so. Patterns also draw on experience to suggest a set of next steps that are likely to make the system yet more Whole. A carefully chosen *sequence* of such patterns—a listing of the

canonical order in which the organization designer introduces change—can resolve product development issues and, importantly, help the organization understand Scrum more in depth.

The Birth of a Pattern

Patterns come from our reflective observations about our hands-on interactions to solve problems in the world. We have the first inkling of a pattern when we see a problem and hear the small voice within us pointing us toward a solution. We try the solution to see if it works in our situation. If it does, then we keep using it and move on to the next challenge. And if it doesn't, we back out the change and try something else. (Sounds agile, doesn't it?) Repeated substantiated experiences with such solutions crystallize them and give them stature as patterns. We may eventually write them down and share them within our community.

Patterns in general come out of community, and the patterns here are no exception. The patterns have their roots in our collective experience. The authors have worked together to choose only the best from a much larger set of candidate patterns that arose along the way. Each pattern has earned its position here because it represents something singular, fundamental, or important, or because it communicates otherwise elusive common sense. The authors have supported each other in reshaping and polishing these patterns to incorporate perspectives from across the whole Scrum community. They reflect input and engagement from a broad constituency and, in particular, from people who carry the foundational insights of Scrum from its inception and early practice.

These patterns go beyond descriptions of vernacular practice to capture the deep nuggets of insight that often elude everyday practitioners. For example, most people believe the purpose of the *¶29 Daily Scrum* is to answer the Three Questions; in fact, the Wikipedia article for stand-up meetings until recently said that the main focus of the *Daily Scrum* was to answer the Three Questions. (Notice that we refer to patterns using a notation that starts with a "¶" semaphore followed by the pattern number and the pattern name. The section Book Notations on page xxi later in this chapter describes such notations and naming conventions in some detail.) To set the record straight, the *Daily Scrum* instead exists as an event where the *¶14 Development Team* replans the *¶46 Sprint*: answering the questions takes only a small fraction of the time and is done just to provide context for the replanning. There are also widespread misunderstandings about what it means to fail a *Sprint*, about the *¶71 Sprint Goal*, and about the role of specification in Scrum. In this book,

we tap into longstanding experience, timeless origins, and the most authoritative sources to set the record straight. Each pattern has been through a long journey of review and refinement so you can enjoy a powerful synthesis of informed and authoritative views on those Scrum elements that they explain and clarify.

Beyond being just a "solution to a problem in a context," each pattern explores the forces at play in the complex contexts of organizational development and workflow. These forces are the *why* behind the pattern. Through these patterns, you can understand the problem in depth and better appreciate *why* the solution might work. However, if you find something in your deepest self, whispering in your ear to try something other than what the pattern suggests, you are probably best to follow that instinct. You know your situation and problem better than we can ever encode in writing. If the pattern awakens your instinct and opens a path towards higher ground, it has done its job. A pattern is not always a goal, but a gate through which you pass on the way to seeing and acting more clearly. And in the agile spirit, the proof is in the tasting. Patterns are often small changes that you can implement provisionally. After inspecting and adapting, you may either go on to the next pattern or may in some cases decide to take one step backwards, undo the pattern, and try something else. Scrum is an empirical framework and each pattern should offer empirical evidence that it works, to validate your decision to apply it.

Pattern Form

We divide written patterns into sections that lead the reader on a literary journey into understanding why the pattern works. A pattern always starts with a picture that serves as a kind of visual mnemonic for the pattern. The first section of text generally describes the situation in which this pattern occurs; this is called the *context* of the pattern. Three stars delimit the end of the context. The second section is a statement of a problem that occurs in this context, usually highlighted in **bold**. Then follow the trade-offs that play in the problem. We call these *forces*, after the metaphor of the forces of gravity and wind that an architect or builder must balance in the built world. However, Scrum patterns, like Alexander's patterns, view forces in a much more inclusive way that includes our vulgar instincts, our aspirations, and our fears. Now we can tie together our context, problem, and forces with a solution. The solution is a simple, direct action in **bold** text, couched in a short explanation to make the solution clear. Then come the three stars again to delimit the end of the solution. In most complex domains there are always loose ends after applying even the best of solutions, so we usually add more

descriptive text detailing how to introduce the solution. This text also describes *how* and *why* the pattern works, so the reader can build on his or her own insight to build the most Whole result. The application of the solution results in a new situation with a new context, with new problems to solve, leading to new patterns.

At the beginning of each pattern, we have annotated the pattern name with zero, one, or two stars. If a pattern has two stars, we believe that you ultimately will need this solution to resolve the forces to move forward in Scrum. If it has one star, we still believe that the pattern is the core of a good solution; however, we cannot argue that it is the *only* way. If the pattern has zero stars we still feel it works as a solution and that it is the best solution much of the time—though other good solutions exist as well.

No Pattern Stands Alone

It's tempting to cherry-pick patterns and try to apply them to solve problems in isolation. In simple systems, we can tire-patch individual problems without being distracted by adjacent concerns. However, in a complex system such as a workplace organization, changes we make in one place may have unintended side effects elsewhere. Though each pattern helps us focus on the problem at hand, context is everything. Applying each pattern in the broader context of the patterns that are already there helps us avoid unintentional setbacks from side effects. A pattern language strives to order patterns in a way that minimizes such setbacks. We have worked hard to do that for you here.

For example, consider the patterns *¶16 Autonomous Team* and *¶10 Cross-Functional Team*. You might decide that team autonomy is such a central agile principle that you want to put it in place early, so you try to apply that pattern first. But then you find that the team really can't be autonomous, as it shows signs of depending on external workers to fill gaps in their expertise as work calls for specific competencies. The commitment to becoming cross-functional needs to be in place before a team can achieve autonomy, so it will be suboptimal to consider one without considering the other.

When you apply a pattern, the solution will itself create new forces that you must resolve at the next level of detail, so it's also important to have hints about what to do next. As much as each pattern describes a single solution to a problem, interdependencies of complex system components imply that no pattern stands alone. We must think of each pattern like the bumper sticker from the 1960s: "Think globally, act locally." Alexander tells us:

> Each pattern can exist in the world, only to the extent that is supported by other patterns: the larger patterns in which it is embedded, the patterns of the same

> size that surround it, and the smaller patterns which are embedded in it. (From *A Pattern Language: Towns, Buildings, Construction [AIS77]*, p. xii.)

We share this fundamental view of the world, that says when you build something you must also repair and make better the world around it. The world at large then gradually improves, becomes more coherent and more Whole. A set of patterns that work together to this end is called a *pattern language*. A language has a grammar that defines "legal" sequences of "parts." In a pattern language, the parts are patterns, but the relationships between the parts are as much or more part of the language than the parts themselves. Each pattern describes the context of those patterns that are prerequisites for the current one. And each pattern also advises us about what other patterns might further refine our Whole to help complete this one. These relationships, or connections, form a structure, a grammar, a language. This book presents two such languages: the Product Organization Pattern Language on page ?, and the Value Stream Pattern Language on page ?.

Sometimes, you just need patience. Fixing an organization often has more of the complexity of a train wreck than that of putting on tire patches. While acting locally, think globally. Sometimes you will need to apply several patterns —maybe over several months—to solve a complex problem in the organization. Together, these patterns will resolve the forces in the organization in a way that allows the solution to emerge. We say that the patterns *generate* the solution, indirectly, rather than subdue the system by force. It might take some mental discipline to subdue your instincts to try to coerce the system into providing immediate gratification. In the end, you can't control a complex system, and it may require several individually small nudges to turn the ship. But remember: apply a single pattern at a time (*¶188 One Step at a Time*).

Patterns Are About People

Most importantly, patterns are about people. Some of our clients want us to tell them what to do in Scrum transformations, or to answer pointed questions about a particular decision in their journey. Those who use patterns in this way are missing the core of their power. It is true that these Scrum patterns encode centuries of hard-won experience. Nonetheless, your particular situation usually brings its own forces, trade-offs, and opportunities—and if you're really in touch with your team and your business, the ability to recognize a great solution already lies deep within you. Rather than creating patterns as instruction manuals, we wrote these to inspire you to carefully consider the forces at play in a particular situation, to draw you into wrestling with problems at a deeper level than business dialectic usually affords, and to lead you

to discover those gems of collective insight that lie latent in the collective experience and spirit of your Scrum Team. They may remind you of what you once knew before you learned the modern techniques of your trade or the practices of current fashion in your industry. Like Scrum, patterns provide no final answers. Both of them are ways to engage our instinct to refine the day-to-day processes by which we build great things that add value to our community.

Kaizen Spirit

Lastly, it will be a long time before you run out of patterns for improving your organization. Scrum, like patterns, is based on a passionate and ever-present spirit of improvement that the Japanese call *kaizen* (カイゼン). A good Scrum Team celebrates the discovery of a new shortcoming, carefully ponders what allowed such a failure to arise, and then looks within itself, perhaps guided by friends and patterns, to seek solutions to the problem. Even when things are going well, great Scrum Teams constantly have the question on their mind: *How can we do even better?* This attitude is the heart of Scrum, and it shows up in events that range from the *Daily Scrum* where the team replans the *Sprint* to squeeze the absolute maximum value from their work, to the *¶136 Sprint Retrospective* where the team collectively reflects on problems and improvement. Both the Value Stream and Product Organization Structure languages offer patterns to make problems visible and effect frequent improvements so the team gets better and better. Value increases accordingly —whether "value" means higher quality product to an end user, or whether it means a better work environment and a happier team.

Getting Started

New Scrum Teams can use pattern languages as a kind of roadmap to introduce Scrum into their organizations, one pattern at a time. Dependencies between patterns form a complex graph, albeit a directed graph with no backtracking. Knowing where to start can be overwhelming. This section will help you get started with pattern languages and pattern sequences.

This book contains two pattern languages. We call them "languages" because they define constraints on the combination and ordering of the parts (the patterns) in the same way that a grammar does (for example, for words). As mentioned earlier, one pattern language (in Chapter 2) deals with the organizational structure and another (Chapter 3) deals with the structure of the Value Stream. Don't let the titles scare you: we'll introduce you to anything you need to know about Value Streams if it's important for moving ahead.

Both pattern languages are broadly about organizational *design*, at a level above the implementation concerns germane to your organization. You can implement each pattern in a million different ways. One language describes the design of the Scrum organization; the other describes the metaphorical stream along which the product flows, turning the raw materials of its tributaries into products that flow into the market. There is usually one Scrum organization per Value Stream.

We split these patterns into two languages because the patterns of each language tend to be highly coupled to each other, but loosely coupled to patterns in the other language. We find that this structuring helps newcomers more readily understand the big picture of Scrum. The connections between patterns became more clear as we added new patterns, and they started to fit together like puzzle pieces that belonged in two separate puzzles. Then we fine-tuned the languages' structure and content by evaluating sequences that they generated, to form a more rigorous "grammar" of ordering. By synchronicity, our overall result matches the division of organizational design patterns found in earlier research such as in *Organizational Patterns of Agile Software Development [CH04]*. As you design your new organization, you should freely pick and choose patterns from both chapters.

To get started, get together as a Scrum Team. Start with the book in one hand and a pad of small stickies in the other. Read through the patterns that catch your eye and put a sticky note on the first page of each pattern that excites you, or makes you nod, or that intrigues you as being able to make your Scrum more whole. Then create a totally ordered list of the patterns you have selected in the order you choose to implement them. Don't get too serious about it; you can follow your intuition and change the ordering at any time. There can be a few patterns from the Product Organization Structure language and then a few from the Value Stream language according to your team's collective insight. To order your patterns, place ones that have more extensive scope (larger context) before those with a more refined scope. For example, if you are a Product Owner and already have a vision for your product, you might start with a Product Roadmap and then introduce a Product Backlog and Product Backlog Items. Maybe you decide to hire someone to help you through the Scrum process from this point on so you bring a ScrumMaster on board. Alternatively, you may hire a Development Team and let it take the lead to hire a ScrumMaster. Eventually you start working towards a Refined Product Backlog. Ordering your list of patterns is always a matter of common sense.

Pattern Sequences

We call this ordered list of candidate patterns a *sequence*, or sometimes a *project language* (which is unrelated to the field of traditional project management, and that the same word is used is merely an accident of history). We present five example sequences in the book: the Product Organization Sequence on page ?, the Value Stream Sequence on page ?, a Product Backlog Sequence on page ?, a A Scaling Sequence on page ?, and a sequence called A Project Language of Highly Effective Teams on page ?. Each sequence describes one typical path through the language, and you can use them as inspiration for your own project language. Most example sequences start at the beginning, assuming you have absolutely no Scrum structure in place; if you have already started, find your existing context in the language and start there. Start with the biggest pattern where you can create something, then move to the patterns that refine it by selecting the smaller patterns you think will apply in your situation; then work through the even smaller patterns that refine these even further, and select the ones that help you fix your particular problem. And thus starts your journey of improvement. Agile is as agile does, and implementing Scrum is itself a process of inspecting and adapting. Implement one pattern at a time so you can unambiguously assess afterwards whether it worked (see *¶188 One Step at a Time*). You'll adjust the patterns in your sequence now and again, reordering, taking some patterns out, and putting others in.

Piecemeal Progress Toward Wholeness

Deciding where to start can be important to gain traction and influence in an organization. But instead of finding "the right starting point" as described earlier, you can simply start anywhere. Taking a pattern from the book and working out how it fits in your organization and implementing the solution will take you on a path to more patterns and more change. Progress will be piecemeal as you move from one pattern to the next. You will find implementing the solution in one pattern will make a change in your situation—in the world—and thus creates a new context and the opportunity to apply another pattern to further improve it. This will be a piecemeal process of changing your team or organization.

Maybe you find that something is missing. We took the patterns to as high a level of excellence as our experience could afford. Maybe you need to finish the job for us, in your particular context. A pattern is always a work in progress. As Paul Valery asserted for his poems, a pattern is never finished by its author: it is only abandoned. We have abandoned them into your hands,

and they will again take up vivacity only when you make them your own. Take the liberty of evolving these patterns in your own direction. Assemble the community and write your *own* patterns, and add them to your own pattern language. We have offered a starting point, and in spite of our experience, we can't foresee every problem that will arise. We certainly don't have an exhaustive knowledge of solutions specific to your situation. Especially in local contexts, you will be able to recall broad prior experience to know what has worked for you before. Such knowledge too often gets lost, and one can often find it hidden among the neurons of the gray-haired folk in your organization. Build a community around this pattern-writing activity and evolve *your* pattern language. You need to build *the* pattern language for your group: each of the book's two major chapters comprises only *a* pattern language. We feel we have roughed out the big picture. We hope you take this accumulated knowledge seriously, but in the end, it's your show.

We want to take this opportunity to implore you to attend to the links each pattern offers: both to the patterns it refines, and to the patterns that further refine it. Although you may be acting locally, everything you do is in the context of the global organization. Always think about the wholeness of the Whole you are striving to improve; about your entire organization inside and outside the Scrum parts; and about all the people whose lives you touch in your world of work.

The Fundamental Process

In the spirit of "build the right process and you'll build the right product," the most basic process is what the agile folk call "inspect and adapt." Deming called it *plan-do-check-act* (PDCA) and later renamed it *plan-do-study-act* (PDSA) to emphasize the importance of analysis over mere inspection. Christopher Alexander calls it *the fundamental process*. This is the heart of Scrum, and the cycle of process improvement is Scrum's heartbeat. The *¶46 Sprint* in Scrum is first a cycle of process improvement and second, a cycle of regular delivery.

The early work of Christopher Alexander (the building pattern guy) on patterns had strong links to Japanese culture and some of its philosophical literature. In the same way, Scrum finds its roots in Japanese industrial culture and its deeper philosophy. In recent years, Alexander expanded his vision of pattern theory, of how to apply patterns, and of how patterns work. In a series of works collectively called *The Nature of Order* (*The Nature of Order: An Essay on the Art of Building and the Nature of the Universe—Book 1, The Phenomenon of Life [Ale04]*, *Book 2, The Process of Creating Life [Ale04a]*, *Book 3, A Vision of a Living World [Ale04b]*, *Book 4, The Luminous Ground [Ale04c]*), Alexander describes the process by which people interact with the built world using

their fullest humanity, striving to increase the Quality in all they do. This process well applies to your Scrum journey. We offer his "fundamental process" as follows as a guideline that might inspire you in your application of these patterns. We include it here as much for the sake of what it says about the importance of the human element as for its steps. For "centers" in the following, you can substitute "patterns." Note that the patterns themselves help guide the intuition of the designer and the steps that the process follows, as Alexander describes in *Book 2, The Process of Creating Life [Ale04a]* (p. 216):

1. At any given moment in a process, we have a certain partially evolved state of a structure. This state is described by the wholeness: the system of centers, and their relative nesting and degrees of life.
2. We pay attention as profoundly as possible to this WHOLENESS—its global, large-scale order, both actual and latent.
3. We try to identify the sense in which this structure is weakest as a whole, weakest in its coherence as a whole, most deeply lacking in feeling.
4. We look for the latent centers in the whole. These are not those centers which are robust and exist strongly already; rather, they are centers which are dimly present in a weak form, but which seem to us to contribute to or cause the current absence of life in the whole.
5. We then choose one of these latent centers to work on. It may be a large center, or middle-sized, or small.
6. We use one or more of the fifteen structure-preserving transformations [simple structures], singly or in combination, to differentiate and strengthen the structure in its wholeness.
7. As a result of the differentiation which occurs, new centers are born. The extent of the fifteen properties which accompany creation of new centers will also take place.
8. In particular we shall have increased the strength of certain larger centers; we shall also have increased the strength of parallel centers; and we shall also have increased the strength of smaller centers. As a whole, the structure will now, as a result of this differentiation, be stronger and have more coherence and definition as a living structure.
9. We test to make sure that this is actually so, and that the presumed increase of life has actually taken place.
10. We also test that what we have done is the simplest differentiation possible, to accomplish this goal in respect of the center that is under development.
11. When complete, we go back to the beginning of the cycle, and apply the same process again.

Sometimes a pattern won't resolve the forces you have in your context. Remember that there are no silver bullets; these patterns are based on the experiences we've collected, and your situation may be different. You *will* need to locally adapt each pattern to *your* situation. You can implement each pattern a million different ways without it ever being the same twice.

A Quick Tour of the Book

This is probably not a book you will read cover to cover. We're glad you're starting with the introduction. A good next step might be to read the very first pattern, *¶1 The Spirit of the Game*: it may help you ponder how high your aspirations should reach when applying these patterns. Then, you might take a day to skim the patterns in the book, reading just the **boldface** text, which we use to help you internalize how the pieces integrate into an overall concept. That's why the boldface bits are there.

The book weaves together two intertwined narratives. The patterns are the heart of the book, and in some sense the rest of the text is secondary supporting material. In accordance with broad convention, each pattern has a number (see Book Notations on page xxi, which follows), and the patterns appear in the book in numerical order. The order of the patterns within Chapter 2 and Chapter 3 reflect a typical order in which one might apply the patterns, the canonical sequence for applying them. We explored this in more detail in Patterns of Scrum on page x. Patterns also refer to each other by number, and the links between them form a graph which reflects the structure of the pattern language that contains them. We have woven the second narrative into spaces between the patterns. It includes chapters such as this one that describe the book's approach, philosophy, and history; a minimal pattern-based exposition of the Scrum Core in Chapter 1; and other small sections that clarify or amplify special topics. Many of these topics, while important to Scrum, are not patterns per se, but instead describe broad concepts or principles on which the patterns stand. So, for example, the section Notes on Velocity on page ? appears right before a group of patterns that focus on velocity. Also among these interludes you will find example sequences that describe how the patterns link to each other in an idealized order of application.

At the heart of the book are the two pattern languages in Chapters 2 and 3. Each of these pattern languages offers one perspective on the multiple forms cutting across your organization. Chapter 2 focuses on the roles, relationships, and gatherings of people, and Chapter 3 on the organization of the workflow. You may certainly pick and choose from both chapters as described earlier. Each pattern is numbered, and patterns refer to each other by pattern number

rather than by page number, in accord with the convention that Alexander used in his work. After reading *The Spirit of the Game*, you might read the main sequences: the Product Organization Sequence on page ?, and the Value Stream Sequence on page ?. Those sequences will give you an overview of the Wholes that the pattern languages represent and may draw you into more deeply reading individual patterns that speak to you.

Chapter 4 talks about how to bring the pattern languages into your own space, and gives an example sequence (project language) to build a high-performing team.

This book builds on past work which itself contributed much to agile foundations, and which itself has a wealth of insight into making Scrum work. Rather than reiterating those patterns here, we refer to the original source. However, we want you to be able to recognize those patterns within yourself when you come across them, and we include short descriptions of each one to help jog your memory. Chapter 5 provides *patlets* (small, one-sentence pattern summaries) for patterns from other sources, particularly from *Organizational Patterns of Agile Software Development [CH04]*. The same chapter contains patlets for the patterns in this book—we tend to use it as an annotated topical index when we are seeking how to solve a particular problem.

Book Notations

Good patterns not only work together as elements of a grammar or language that can generate countless system variants, but their names also enrich the design vocabulary in the everyday spoken and written language of organization design discourse. So the term *Sprint Review* is three things. First, the *Sprint Review* is the name we give to a group of people who assemble for a time-boxed interval each *¶46 Sprint* to assess the state of the product increment. We can point to that group of people and say, “Oh, look, they’re having their *Sprint Review* over there right now.” You can see it, point to it, and refer to any instance of it by name. Second, it describes the process one uses to create such a gathering, and the process can be found in writing under the name *¶35 Sprint Review* in this book. You notice that the name begins with the curious prefix *¶35*, to be read: *Pattern 35*. The patterns appear in the book in numerical order according to this prefix. Lastly, *Sprint Review* becomes a term that we use in our everyday language as we discuss our Scrum implementation, how it is going, and how to improve it.

This book frequently uses pattern names in this latter sense, as though they were ordinary English phrases. We try to introduce patterns in the book before referring to them in this way. But, still, we call out each use of such terms

as a pattern, so as to distinguish the words from vernacular English. The traditional pattern literature dating back to the late 1970s has always distinguished pattern names in a small caps typeface. However, since there were technical issues that prevented us from using the preferred typeface, our work-around is to distinguish patterns with a prefix that includes a ¶ character and the pattern number as described earlier. But since a page full of such prefixes (and for those using electronic books or reading this on the web, a page full of hyperlinks) could potentially be too distracting, only the first reference to a given pattern within a given section will appear this way; subsequent references appear in a *distinguished typeface*. The book often refers to organizational patterns from other sources, though the section Patlets on page ? includes a synopsis of each one called a *patlet*. When we reference one of those patterns, we give its name, its pattern number (with the perfunctory ¶) and a reference (page number or hyperlink) to the patlet text within the book. For example, we would refer to *Community of Trust* as *¶95 Community of Trust* on page ?.

What Now?

We've covered a lot of ground in this introduction; congratulations for getting this far. Next up is the first chapter of the book, The Scrum Core as Patterns on page ?. In this chapter, you get better acquainted with Scrum, starting with our first pattern (*¶1 The Spirit of the Game*) which is followed by a definition of Scrum using the patterns in this book.

As we wrote earlier, you do not have to read the book cover to cover but we do recommend you read the first chapter. *The Scrum Core as Patterns* helps to familiarize you with what may be new to you (patterns) through something familiar (Scrum). However, you don't have to start there. Here are some other options you might like to try.

- Read through one of the sequences. They give you an overview of the patterns in the language and one way to order their use.
- Skim all the patterns by reading just those parts of the patterns that appear in boldface.
- Start with an element of Scrum your team is struggling with, read that pattern and the patterns that refine it, then use these to improve how you are working.
- Find a pattern that matches where your teams are and see what refines this position in the pattern languages.

- Read through the Patlets on page ?, which are short summaries of all the patterns in the book (and in a few other books).
- Flip through the book and stop at a pattern that looks interesting or has an interesting title (*¶20 Oyatsu Jinja (Snack Shrine)* usually grabs attention).
- After you have spent time reading, reflecting, and trying the patterns, we invite you to read Composing Your Own Pattern Language on page ?.

No matter how you start, we do recommend having a pencil and some sticky notes at hand. Take advantage of the whitespace in the book to write your thoughts. Use the sticky notes to mark pages you want to come back to. We know when a book has had an impact on us because it is worn from many readings; we hope that is the case for your copy.