Extracted from:

A Scrum Book

The Spirit of the Game

The Pragmatic Bookshelf
Raleigh, North Carolina

The Pragmatic Programmers

A SCRUM BOOK

THE SPIRIT OF THE GAME

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

The team that produced this book includes:

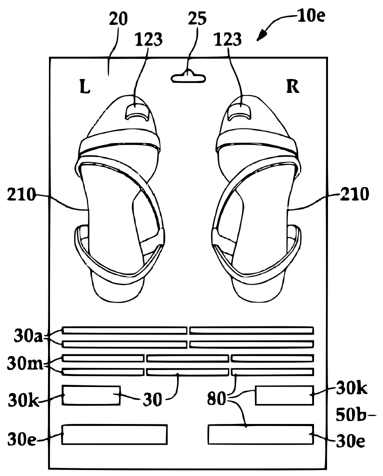
Publisher: Andy Hunt
VP of Operations: Janet Furlow
Managing Editor: Susan Conant
Development Editor: Adaobi Obi Tulton
Copy Editor: Sean Dennis
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics

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

¶63 Enabling Specification



Enabling specification for U.S. Patent 7,329,448: Adhesive Pads for Footwear.

...the *¶54 Product Backlog* is in place and the *¶11 Product Owner* is working on *¶55 Product Backlog Item*s (*PBI*s).

✣ ✣ ✣

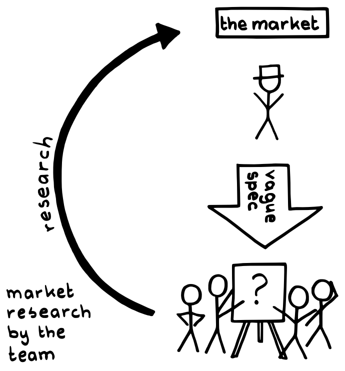
Unexplored requirements cause unpleasant surprises.

The agile tradition holds that user stories suffice as requirements artifacts, and early agile practice often blindly believed in deferring decisions and in having a ready, at-hand, on-site customer who could compensate for requirements shortfalls discovered during development. Given this background, it's easy for the *Product Owner* to throw ideas into the *Product Backlog* and think "that's good enough." Scrum can even support the *Product Owner* doing this because the *Product Backlog* is the *Product Owner*'s artifact, so we assume he or she can call the shots. We also welcome emergent requirements in Scrum, assuming the *Product Backlog* will change as everyone connected with the product learns more, and this may also lead us to having poorly considered *PBI*s. Having the *Product Owner* collocated with the team is often assumed to obviate this problem, but a half-baked idea is still a half-baked idea even if we socialize and review it.

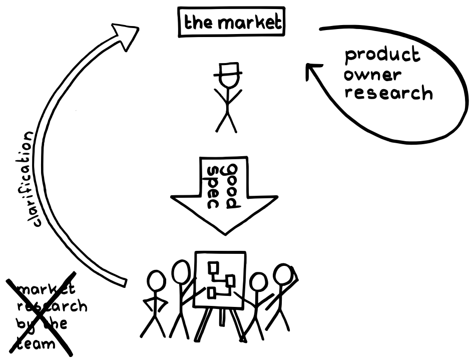
Because Scrum acknowledges emergent requirements, it's easy for the business to claim that the real requirements will come out only when pushing on them during design. On the other hand, full-blown development is a heavy-handed way to elicit requirements. Sometimes all it takes is a bit of dialog between the *Product Owner* and the developers, or between the end users and the *Product Owner*. Testers are notoriously good at sniffing out requirements lapses—lapses that are often easy to fill in (by talking to end users or other stakeholders) once they are discovered. And the further you get into development, the more difficult it becomes to engage end users.

So it's good to do some up-front discovery, exploration, and planning, and to socialize proposals and open issues across all stakeholders. Scrum's roots in Japanese manufacturing also bear out this focus on engagement and planning. But how much should be communicated from the *Product Owner* to the *¶14 Development Team*, and how? A user story isn't even a full requirement: it is just a statement of some stakeholders' identities and their motivation behind some parts of what the system provides. They're just one small part of "the requirements." And while requirements help articulate a *PBI*'s contribution to the *¶39 Vision*, a requirement is not a definition of a product increment—it is only one insight on how that increment meets some isolated want or need. After all, it's a *Product Backlog* and not a "requirements backlog."

Most schedule surprises come from misunderstandings with roots in poor or poorly communicated analysis. Much of Scrum depends on the team having a stable Notes on Velocity on page ?. Once a system is designed, implementation effort usually can be predicted with a high degree of confidence. Once analysis is complete, design and implementation can often proceed without too many surprises. Since estimation focuses on what happens within the *¶46 Sprint*, it's important to move the uncertainty of analysis outside the *Sprint*—into the *Product Owner* process. If you don't do this, at least part of the task of analysis will fall to the developers. That greatly slows velocity, and because the team is not expert in analysis or end-user and market perspectives, the requirements suffer as well:



***Therefore:* The *Product Owner* should deliver *Enabling Specification*s as a sign that he or she has done due diligence in exploring the requirements space**. An *Enabling Specification* is a specification rich enough that someone reasonably skilled in the discipline can implement a solution without substantial subsequent clarification with people outside the *¶7 Scrum Team*. The *Scrum Guide*[20] says that part of the job of the *Product Owner* is "[e]nsuring the Development Team understands items in the Product Backlog to the level needed."



The *Product Owner* and *Development Team* work diligently together to bring the top of the *Product Backlog* to *Ready* (see *¶65 Definition of Ready*) in the weeks leading up to *¶24 Sprint Planning*, exploring all opportunities to create mutual understanding. But in the end, the *Product Owner* is responsible for bringing the *Development Team* enough information so it has a clear understanding of what to build. The *Development Team* members have the final say over whether they will take a *PBI* into the *Sprint* for implementation, based on whether they are confident in their understanding of the *PBI*.

✣ ✣ ✣

Enabling Specification in hand, the *Development Team* is prepared to create a *¶72 Sprint Backlog* based on a deep understanding of the upcoming delivery stream. If the *Development Team* collectively do not believe that the *Product Owner* has communicated enough information to enable them to succeed in delivering the item, the *Development Team* will not pull the *Product Backlog Item* into a *Sprint*.

Establishing this understanding before the *¶75 Production Episode* starts helps the *Development Team* stay in "flow" without having to interrupt development with the distraction of a meeting with the *Product Owner*. In the

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

end, the specification must live in the collective mind of the *Development Team*; that something in a document meets some criteria of completeness is almost immaterial. The specification in the developers' minds becomes enabling through interaction, explanation, prototypes, examples, and an exchange of questions and answers. A great *Enabling Specification* includes written test criteria so that test development is straightforward, and so there are objective criteria to support a ship/no-ship decision at the end of the *Sprint*. If the *Enabling Specification* isn't good enough to specify how the product increment will be tested, it isn't good enough to give the team honest confidence in how to build it.[21]

Of course, there are limits to perfection. If the *Development Team* finds itself in doubt about a *Product Backlog Item* while working on it during the *Production Episode*, then the *Development Team* and *Product Owner* should strive to resolve the gap at the earliest opportunity. The same applies if emergent requirements arise.

It is less important that the specifications be written down beforehand than it is that the *Product Owner* has done his or her homework and that the team has thoroughly discussed the new item. A written *Product Backlog Item* can memorialize and testify to the extent of that research. In fact, much of this research perhaps entailed discussions with the *Development Team*, suppliers, and partners, as well as market research, customers, and of course, end users.

A single, passionate message and written *Product Backlog Item*s alone aren't enough. Research shows that managers are better respected and get their message across more effectively if they deliver it multiple times, often through different media, relying on redundancy to drive the message home. A good *Product Owner* will mix informal descriptions, user stories that underscore the user motivation, use cases that explore edge cases (*Lean Architecture for Agile Software Development [BC10]*, pp. 167–169), visuals that preview the user experience, prototypes that encode major expectations, and anything else that may open its own path to understanding, complementing the others. *Harvard Business Review 89 [NL11]* The *Product Owner* should socialize new items with the team as soon as they arise, at the earliest opportunity when they and the team are working toward a *¶64 Refined Product Backlog*. Each *PBI* can become more and more enabling at each backlog refinement effort and at each *Sprint Planning* event.

21. Jeff Sutherland. "Enabling Specifications: The Key to Building Agile Systems." ScrumInc.com, https://www.scruminc.com/enabling-specifications-key-to-building/, 2 June 2012 (accessed 5 January 2017).

The phrase *enabling specification* is a term of law, applied as a standard for valid patents:

> "A patent specification is enabling if it allows a person of ordinary skill in the art to practice the invention without undue experimentation." *Intellectual Property Law: Commercial [DM91]*

A valid patent must serve to enable anyone reasonably skilled in the general area of the patent to be able to reproduce the invention from information in the patent alone without consulting the inventor. Analogously, the information that the *Product Owner* conveys to the team (whether in writing or not is immaterial) before the start of the *Sprint* should carry them through the *Sprint* in most instances, without need for further face-to-face consultation.