Extracted from:

Lean from the Trenches

Managing Large-Scale Projects with Kanban

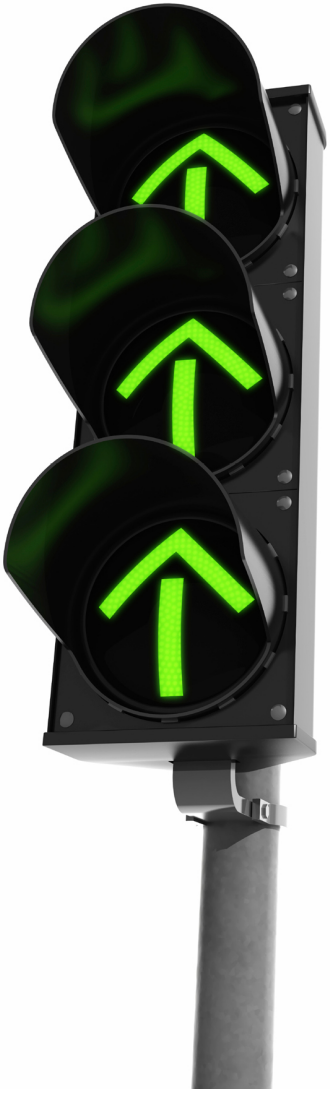
The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

Lean from the Trenches

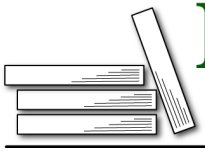
Managing Large-Scale Projects with Kanban



Henrik Kniberg

Foreword by Kent Beck

Edited by Kay Keppler



Pragmatic Bookshelf

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at *http://pragprog.com*.

The team that produced this book includes:

Kay Keppler (editor)
Potomac Indexing, LLC (indexer)
Kim Wimpsett (copyeditor)
David J Kelly (typesetter)
Janet Furlow (producer)
Juliet Benda (rights)
Ellie Callahan (support)

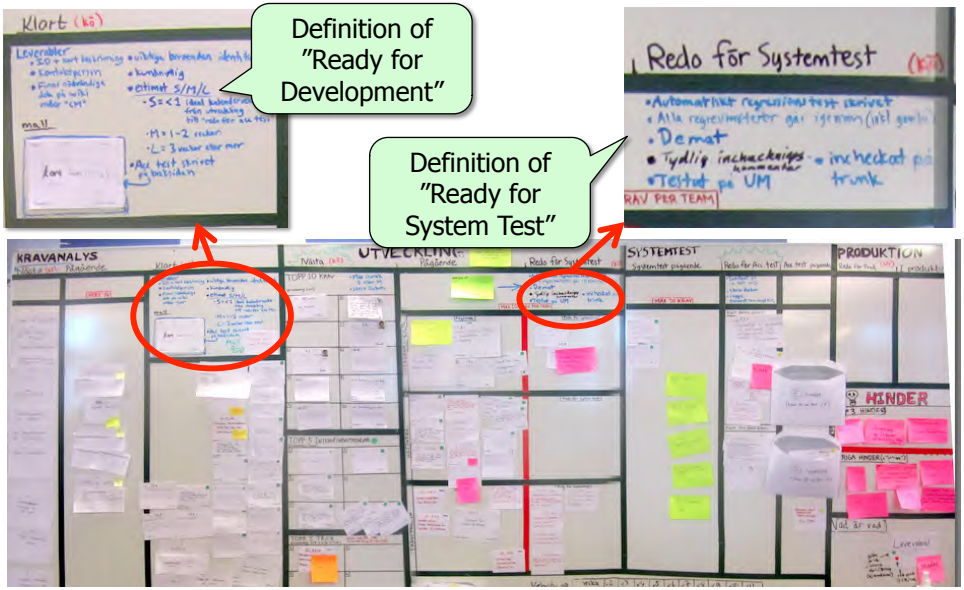
Printed in the United States of America.
ISBN-13: 978-1-934356-85-2
Printed on acid-free paper.
Book version: P1.0—December, 2011

CHAPTER 7

Defining Ready and Done

It's important to be very clear about what the columns on the board mean. Especially in big projects, the more people involved, the greater the risk of confusion, and the greater the *cost* of confusion.

The blue text at the top of most columns on our project board is the *definition of done* for that column (which also means *definition of ready* for the subsequent column). The two most important definitions for us are *definition of ready for development* and *definition of ready for system test*, since that's where we used to have the most problems.



7.1 Ready for Development

The "Ready for Development" column essentially means "Here's a bunch of features that we've broken down and estimated and clarified, but we haven't yet decided which of these we are going to develop and in which order." So, this corresponds roughly to a Scrum product backlog. For a feature to be ready for development, it must have the following characteristics:

- It must have an *ID*. The ID is used as a key when you're looking up more information about this feature, in case there are any associated use case specifications or other documents. These documents are accessible on the project wiki by clicking the corresponding ID.
- It must have a *contact person*. The contact person is typically the requirements analyst who has the most domain knowledge about this feature.
- It must be *valuable* to customers. When breaking down epics into deliverable stories, we want to make sure we haven't lost the customer value along the way. The requirements analysts have the final say on this matter.
- It must be *estimated* by the team. The estimates are normally done by a small group consisting of a tester, a developer, and a requirements analyst playing Planning Poker (see Chapter 19, *Sizing the Backlog with Planning Poker*, on page ?). We use T-shirt sizes (small, medium, large). These are size estimates, not time estimates. But to make the estimation process easier, we use this as a rough guideline:
  - *Small* means "Under perfect conditions this will take less than one week of elapsed time to get to 'Ready For Acceptance Test.'" *Perfect conditions* means that we have exactly the right people working only on this feature with no disruptions.
  - *Medium* means one to two weeks (again, under perfect conditions).
  - *Large* means more than two weeks. Large features have to be broken down further before they are allowed into development.
- It must have an *acceptance test* scenario written on the backside of the card. This is a concrete set of steps describing the most typical test scenario. Here's an example:

  "Joe Cop logs in, looks up case #235, and closes it. He then looks up case #235 again and sees that it's closed."

7.2 Ready for System Test

"Ready for System Test" means that the feature team has done everything they can think of to ensure that this feature works and doesn't have any important defects. They have, however, focused on testing the feature itself, not the whole release that it would be part of.

For a long time system test was a bottleneck, and one of the major reasons for that was the high number of unnecessary defects passing into system test. By "unnecessary defects," I mean feature-level defects that could have been found way before putting it all together into a system test. So, our *definition of ready for system test* is there to keep the quality bar high and catch those pesky bugs early. It's also there to give the feature team a sense of responsibility for quality and to give them permission to spend the necessary time to ensure that a feature really works, before delivering it to system test and moving on to the next feature.

So, here is our *definition of ready for system test*:

- *Acceptance test automated*: This means that some kind of end-to-end feature-level acceptance test or integration test has been automated. We used to use Selenium for that (which runs tests directly against the web interface), but we eventually moved to Concordion. The Selenium tests were just too brittle for our Ajax-riddled web interface, and Concordion fit better with our move toward Specification By Example.[1]
- *Regression tests pass*: All automated tests for previously existing features pass. Sometimes a new feature breaks an old feature, so we have to make sure that all old tests are run on a regular basis.
- *Demonstrated*: The team has demonstrated this feature to the rest of the team, the on-site user, the requirements analyst, the system tester, and the usability expert. This helps us catch usability issues *early* so they don't show up in system test or (even worse) user acceptance test.
- *Clear check-in comments*: When checking in code related to this feature, the check-in comment should be tagged with the ID of this feature, plus an easily understandable comment about what was done. This provides a minimum level of traceability (big projects always seem to fuss about traceability...).

1. www.specificationbyexample.com

- *Tested in the development environment*: Each team has a dedicated test environment, and this feature should be tested there (to avoid the "Hey, it works on *my* machine" syndrome).
- *Merged to trunk*: Code for this feature should be on the trunk, and any merge conflicts should be resolved. This is the basis of the stable trunk model we use (see Chapter 14, *How We Do Version Control*, on page ?).

7.3 How This Improved Collaboration

These two policy statements—*definition of ready for development* and *definition of ready for system test*—have significantly improved collaboration between the teams. This improvement stood out clearly when I did a short survey to check what people thought about all the process changes so far.

In the past, when we just started doing Kanban, each specialty team focused mostly on "their" part of the project board. The requirements analysts looked only at the left part of the project board and considered themselves "done" with a feature when a requirements document had been written. The developers looked only at the middle of the board, and the testers looked only at the right. The testers weren't involved in writing the requirements, so once a feature reached test, there was often confusion about how it was supposed to work. People spent a lot of effort arguing about the level of detail needed in the requirements documents.

These were just old habits. But the project board helped everyone *see* the problem, which is the first and most critical step toward solving it!

The collaboration problems gradually disappeared (well, significantly declined at least) within a few weeks after everyone had agreed on the definitions. The *definition of ready for development* can be achieved only if all specialties work together to estimate features, to break them into small enough deliverables without losing too much customer value, and to agree on acceptance tests.

Similarly, the *definition of ready for system test* can be achieved only if all specialties work together to run feature-level tests (both automated tests and manual exploratory tests) to determine whether this feature is good enough to release.

This clear need for continuous collaboration is what made the test team and requirements team agree to "lend" specialists to each feature team, thus making each feature team truly cross-functional (and *much* more effective)!

In general, writing a *definition of ready* at the top of each key column is one of those simple techniques that is useful in any kind of Kanban system.