Extracted from:

Lean from the Trenches

Managing Large-Scale Projects with Kanban

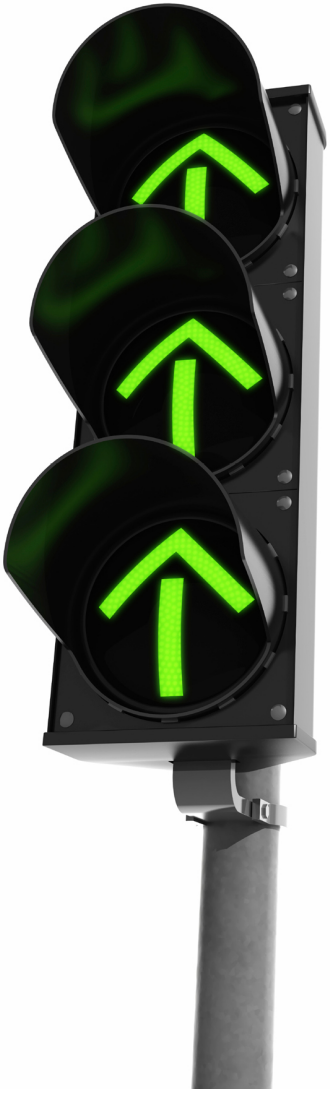
The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

Lean from the Trenches

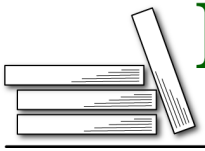
Managing Large-Scale Projects with Kanban



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)

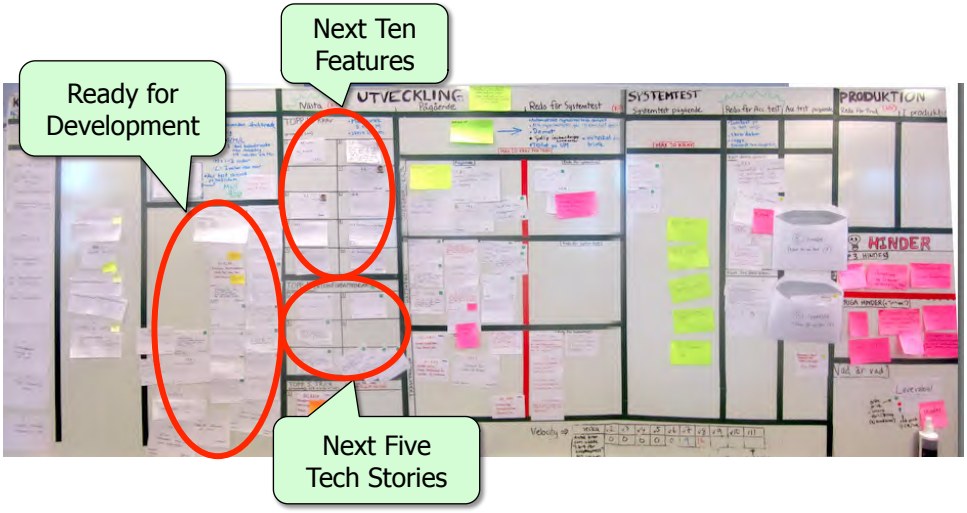
Printed in the United States of America.
ISBN-13: 978-1-934356-85-2
Printed on acid-free paper.
Book version: P1.0—December, 2011

CHAPTER 8

Handling Tech Stories

Tech stories are things that need to get done but that are uninteresting to the customer, such as upgrading a database, cleaning out unused code, refactoring a messy design, or catching up on test automation for old features. We call these *internal improvements*, but in retrospect, *tech stories* is probably a better term, since we're talking about stuff to be done with the product, not process improvements.

Tech stories are born in the "Ready for Development" section of the project board and enter development through a "Next Five Tech Stories" section (right below "Next Ten Features"). These are essentially two parallel input queues to development.

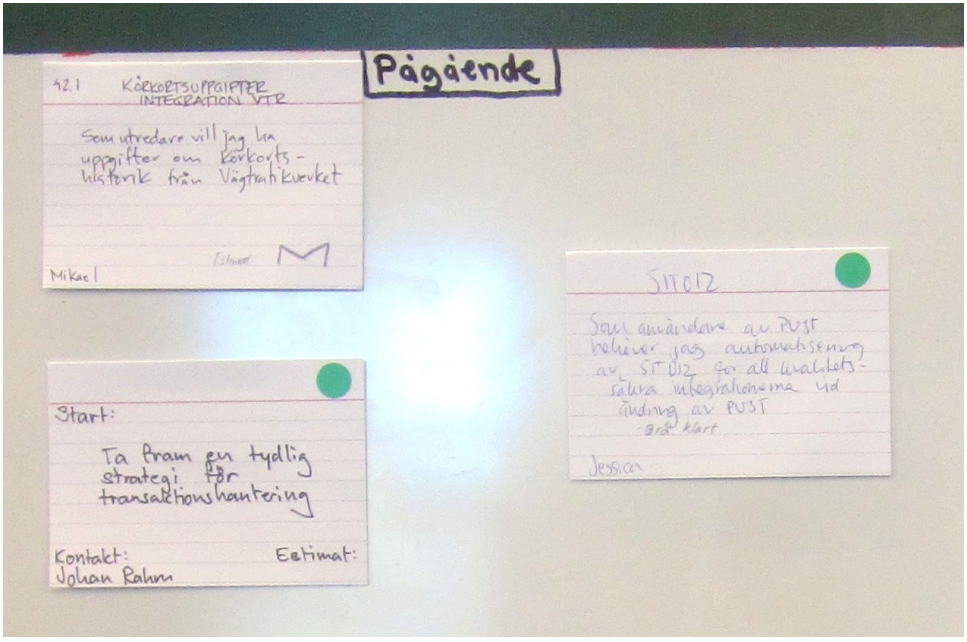


As you can see, there are quite a lot of cards under "Ready for Development," a mixed bag of features and tech stories. We don't waste time trying to put all those in priority order. Instead, we do a form of "just in time" prioritization

by continuously identifying the next ten features and next five tech stories. This provides just enough of a work buffer to keep the feature teams from running out of things to work on.

When a feature team has capacity to start something new, they either pull a feature from “Next Ten Features” or pull a tech story from “Next Five Tech Stories.” We have no static rule defining the correct balance between these two. Instead, we continuously discuss and adjust the balance during the daily stand-up meetings.

Tech stories are distinguished from features by a green spot in the corner of the card. This lets us distinguish between the two even after they have been pulled into development, so the project board reveals how we are balancing our time between features and tech stories.



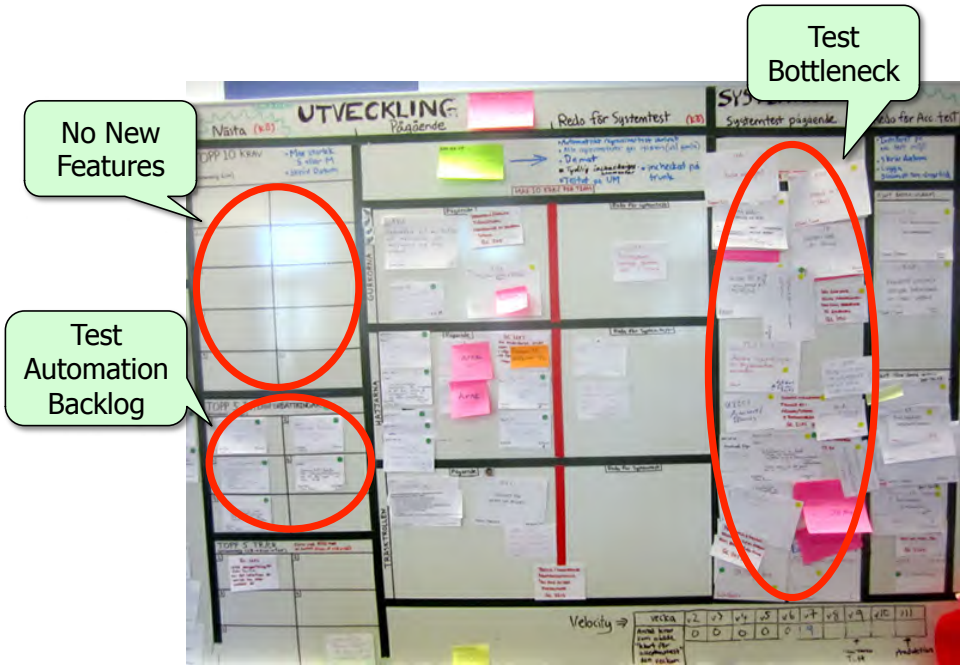
Usually features dominate, but here are some examples of situations that caused us to focus mostly on tech stories for a while:

8.1 Example 1: System Test Bottleneck

System testing had become an obvious bottleneck, so there was clearly no point developing new features and adding to the bottleneck. Once this became clear, the developers focused on implementing tech stories that would make system test easier—mostly test automation stuff. In fact, the test manager

was tasked with creating a *test automation backlog*, prioritizing it, and feeding it to the developers via "Top Five Tech Stories." The testers became customers!

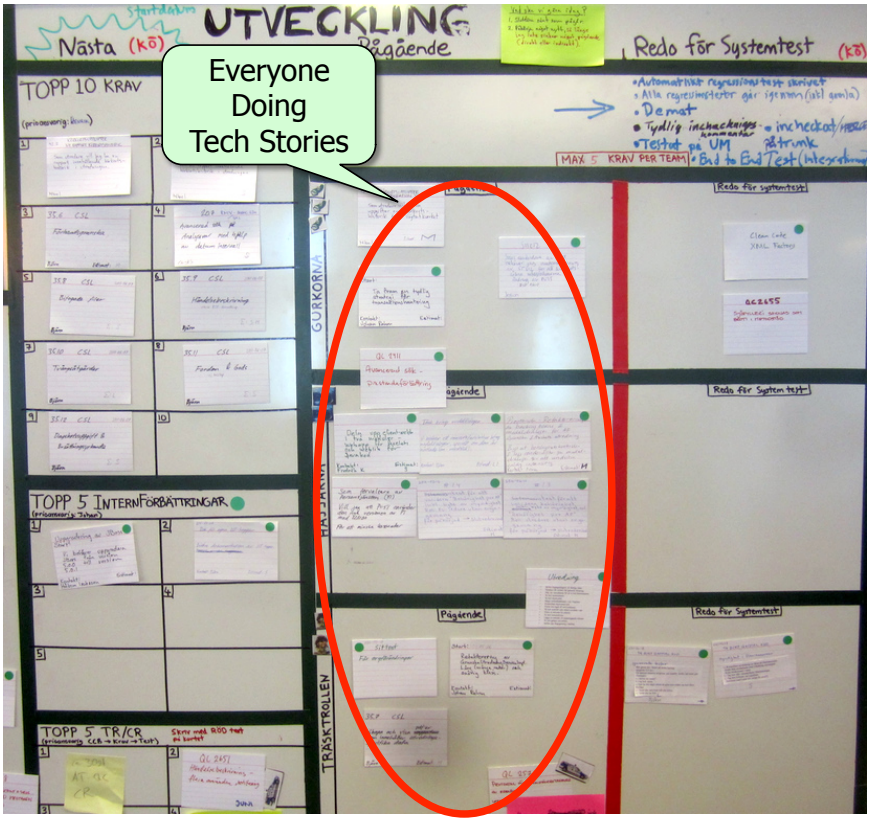
For more information on this technique, see Chapter 18, *Reducing the Test Automation Backlog*, on page ?.



8.2 Example 2: Day Before the Release

It was the day before a major release, and the team wanted to get that release out the door before starting a bunch of new features. So, they focused on last-minute bug fixing. If they didn't have any bugs to fix at the moment, they worked on tech stories—typically things that we had wanted to do for a long time but had never gotten around to, such as removing unused code, catching up on refactoring, and learning new tools.

As you see from the board, lots of tech stories (green spots) are in progress.



8.3 Example 3: The 7-Meter Class

Here's a cool way to make a business case for a tech story. One of the classes in our code base was getting way out of control and needed some significant refactoring, but there was some resistance to spending time on that. So, one of the team leads *printed out the whole class* and laid it across the conference table! It was more than 7 meters long (23 feet)!

Looking at that monstrous printout, everyone clearly saw that we needed a tech story to fix that class immediately! No argument needed. This also illustrated the consequence of being in a hurry and not paying enough attention to design.

We had some fun speculating about future developments along this theme. How about if we estimate features in code-meters and measure velocity in code-meters per day? We could even separate *ideal* code-meters (how long the code would be if we kept it really clean), with *actual* code-meters. Subtract those two, and you get technical debt—in meters! We could even draw a line in the floor to symbolize how much technical debt we have ("Hey look, we

On a Side Note...

I've never seen a project of this scale with so little *drama* in conjunction with releases! Almost disappointing....

Where is the customary panic and rush and all-night crunching the day before the release? Where is the subsequent onslaught of support issues and panicky hot fixing the day after the release? I came in the day after the most important release (the nationwide release that was the focal point of the whole project), and there was barely any sign that anything significant had happened.

The reason for this was that the releases were well-rehearsed, because of the setup with on-site users and pilot releases. Of course, we'd had some problems with the earlier pilot releases—but that's why we do pilots, right?

Anyway, remember to celebrate releases—even when you get good at it and they're not as exciting anymore.

have 23 meters of debt!"). Or maybe we would have to use code-miles for that....

Um, OK, I'll stop now.

Anyway, now that we've started talking about code quality, let's talk about bugs.