

Extracted from:

Lean from the Trenches

Managing Large-Scale Projects with Kanban

This PDF file contains pages extracted from *Lean from the Trenches*, published by the Pragmatic Bookshelf. For more information or to purchase a paperback or PDF copy, please visit <http://www.pragprog.com>.

Note: This extract contains some colored text (particularly in code listing). This is available only in online versions of the books. The printed versions are black and white. Pagination might vary between the online and printer versions; the content is otherwise identical.

Copyright © 2010 The Pragmatic Programmers, LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

Lean from the Trenches

Managing Large-Scale
Projects with Kanban

Henrik Kniberg

Foreword by Kent Beck

Edited by Kay Keppler





Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and The Pragmatic Programmers, LLC was aware of a trademark claim, the designations have been printed in initial capital letters or in all capitals. The Pragmatic Starter Kit, The Pragmatic Programmer, Pragmatic Programming, Pragmatic Bookshelf, PragProg and the linking *g* device are trademarks of The Pragmatic Programmers, LLC.

Every precaution was taken in the preparation of this book. However, the publisher assumes no responsibility for errors or omissions, or for damages that may result from the use of information (including program listings) contained herein.

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)

Copyright © 2011 The Pragmatic Programmers, LLC.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

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

Attending the Daily Cocktail Party

If you walk into this project on any day before 10:15 a.m., it will feel like walking into a cocktail party! People are everywhere, standing in small groups and communicating.



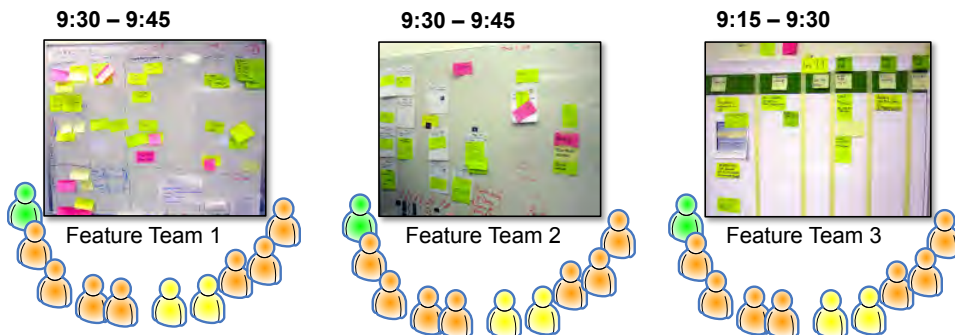
You'll see groups standing in semicircles in front of boards, engaged in conversation, and moving sticky notes around. You'll see people moving between teams, debates going on, and decisions being made. Suddenly a group will break apart, and some individuals will move to another group to continue the conversation. New groups sometimes form in the hall to follow up on some side conversation.

By 10:15 the daily cocktail party is over, and most people are back at their desks.

This may look chaotic at first glance, but in fact, it's highly structured.

3.1 First Tier: Feature Team Daily Stand-up

First up are the feature team daily stand-ups.



Two of the teams meet at 9:30, and one of the team meets at 9:15 (each team decides their own meeting time). Everyone on the team stands up in a rough semicircle in front of their task board, discussing the work they are going to do today and any problems and issues that need to be addressed.

Sally: *I'm going to chase that darned memory leak today.*

Jeff: *You probably need to upgrade the profiler tool first. I had problems with that last week.*

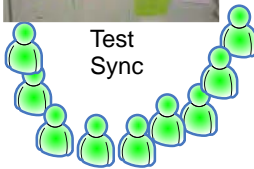
Sally: *OK, thanks for the heads-up. I'll come get you if I get stuck.*

Some teams use the Scrum formula (answering “What did I do yesterday,” “What am I doing today,” and “What is blocking me”), and others are more informal about it. These meetings usually take ten to fifteen minutes and are facilitated by a team leader (which equates pretty much to Scrum master).

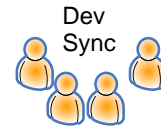
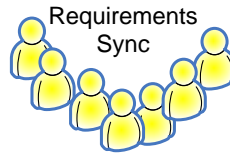
3.2 Second Tier: Sync Meetings per Specialty

At precisely 9:45, a second set of daily stand-ups takes place, during which the members of each specialty (requirements analysis, test, development) meet separately to synchronize their work across all feature teams.

9:45 – 10:00



9:45 – 10:00



All the testers gather in front of a test status board and discuss how to make best use of their time today. The embedded testers have just completed the daily stand-up within their feature team, so they have fresh information about what is going on within each team.

Tom: *Today we need to focus on usability issues in system test. Any help is appreciated.*

Lisa: *I'll join you in an hour or so. My team is finishing a logging feature. After that, they can probably do without me for the rest of the day.*

At the same time, the requirements analysts are having their own sync meeting, including the embedded analysts who just came out of their feature team stand-up meeting with fresh information.

Jim: *The folks on my team seem confused about the new usability guidelines.*

John: *My team too!*

Maria: *Oh, maybe that's why system test has become a bottleneck again. They seem to be struggling with inconsistent user interface design. Any proposals?*

Jim: *Let's set up a workshop and discuss the new guidelines.*

Maria: *OK, I'll bring up this at the project sync meeting right after this. We'll find a good time today and try to get at least one developer and tester from each team to join.*

At the same time, the team leads from each feature team, plus the development manager are having their *dev sync meeting*. The team leads just came out of their feature team stand-up meeting with fresh information.

Jeff: *My team is finishing off the logging feature (points to a card on the wall). We'll probably get started with the database migration this afternoon.*

Sam: *Wait, does that mean we need to update the build scripts?*

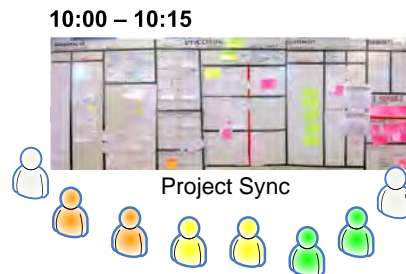
Jeff: *Yeah. It's easy, though. Ask Lisa if you need help. At the team stand-up, she said she didn't have much to do today.*

The test sync meeting takes place in front of a test status board, while the requirements sync and dev sync meetings take place in front of the project board (see [Chapter 4, The Project Board, on page ?](#)). These three meetings take place in parallel just a few meters from each other, which makes it a bit noisy and chaotic, but the collaboration is very effective. If anybody from one team needs info from another, they can just walk over a few meters to the other meeting and ask a question.

Some people (such as the project manager and I) float around between the meetings, absorbing what is going on and trying to get a feel for which high-level issues need to be resolved. Sometimes we stay outside the meetings, and sometimes we get pulled into a discussion.

3.3 Third Tier: Project Sync Meeting

Finally, at precisely 10 a.m., the project sync meeting takes place in front of the project board.



The people at this meeting are informally referred to as the *cross-team* (or *tvärgrupp* in Swedish)—a cross-section of the whole project. In our case, that equates to one person from each specialty and one person from each feature team, plus a few other folks such as the project manager, configuration manager, and myself.

The project sync meeting is where we look at the big picture, focusing on the flow of functionality from analysis to production: Which team is doing what today? What is blocking our flow right now? Where is the bottleneck, and how can we alleviate it? Where will the bottleneck be next? Are we on track with respect to the release plan? Does anybody not know what to do today?

This not only gives us a great bird's-eye perspective of what is going on, it lets us solve problems quickly, especially collaboration issues between teams. If “us” and “them” work together every day, then sooner or later “us” and “them” become just “us.”

That's it. A total of seven stand-up meetings every day, organized into three layers. Each meeting is *timeboxed* to fifteen minutes, each meeting has a core set of participants who show every day, and each meeting is public, so anybody can visit any meeting if they want to learn what is going on or have something to contribute. And it's all over by 10:15 a.m.

If some important topic comes up during a daily and can't be resolved within fifteen minutes, we schedule a follow-up meeting with the people needed to resolve that issue. Some of the most interesting and valuable discussions take place right after the project sync meeting, as people stand in small clusters dealing with stuff that came up during the daily stand-ups.

This structure of daily meetings was something that we gradually evolved into. When we started doing the “daily cocktail party” (which, by the way, is my term, not an official term we use in the project), I was a bit concerned that people might think we were having too many meetings. That turned out not to be the case. On the contrary, the team members insist that these meetings are highly valuable, and I can see that the energy level is usually high and problems get solved.

Most people need to go to only one meeting. Some individuals need to go to two meetings. The team lead of a feature team goes to his team stand-up as well as the dev sync meeting. The embedded tester in a feature team goes to the team stand-up as well as the test sync meeting, and so on. This is a very effective way of “linking” communication channels and making sure that important knowledge, information, and decisions propagate quickly throughout the entire project.

Many problems that would otherwise result in the creation of documents and process rules are resolved directly at these morning meetings. One concrete example is deciding which team is to develop which feature; another example is deciding whether to spend our time developing customer-facing functionality today or spend it implementing customer-invisible improvements to the technical infrastructure. Instead of setting up policy rules for this, the teams simply talk about this during the daily meetings and make decisions on-the-fly based on the current situation. This is the key to staying agile in a big project and not getting bogged down in bureaucracy.