Kanban – some notes

Chris Baker

Project Management Limited

Version 2, August 2009[1]

Kanban is a progress-monitoring method that was developed in "just in time" manufacturing (especially at the Toyota motor company). It is now being adopted as a project management tool for software projects. It creates a visual display of how the project is progressing, thus allowing progress to be tracked easily and actual or potential bottlenecks to be dealt with. It also has rules intended to reduce the amount of work in progress. It values smooth flow of work through the project, focusing on quickly removing impediments and talking root causes.

Kanban is being adopted particularly by projects that use Agile methods, as it goes nicely with the agile manifesto aim of simplifying and reducing processes, tools, documentation, and plans. However, kanban could readily be used in the production-line-like situations that occur in many projects, regardless of the project management methodology the project is using.

I have been studying the explanation of kanban at the PM podcast website and wanted to work through an example of my own to cement my understanding. This is my example.

Kanban example

Imagine that I am managing a project. The project has been divided into creating a number of features (these could be features, or user stories, or any other sensible units of work).

The project involves taking each feature through a sequence of steps.

1. Design
2. Coding
3. Testing
4. Fixes

In planning we have estimated that each feature will spend a day at each step, and we expect that each step in the production line can work on 3 features at a time.

[1] This version of the article was updated in August 2009, following very helpful criticism of the initial version by David Anderson, Author of "Agile Management for Software Engineering" and a good kanban blog. I'm very grateful to David for his input – especially on the importance of limiting work in progress, which I had missed. Any remaining shortcomings in the article are of course entirely my responsibility!

Setting up

We are set to begin the production line on Monday. Most of the team share an office which has a large notice-board. On the Friday before we start the production line, I clear this off, and divide it into 5 columns

1. Design
2. Coding
3. Testing
4. Fixes
5. Ready

Each column is divided into two – an upper section for work in progress, and a lower section for work that has completed the current stage and is ready to enter the next stage

We make a set of index cards, one for each feature. Each card has the feature's name, and perhaps a brief description. The card is the "kanban" for that feature (kanban is a Japanese word meaning sign or token). We are going to move the cards across the board to represent how the features are progressing through the production line.

Monday

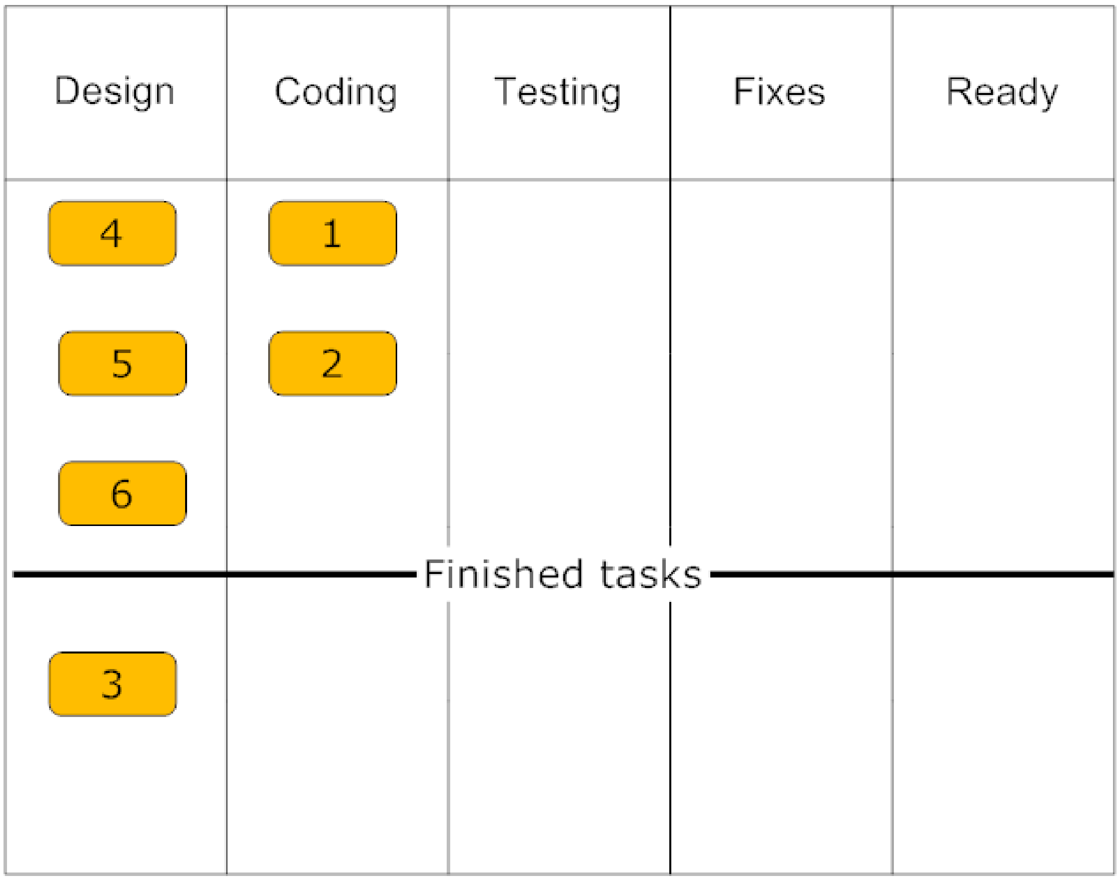

Figure 1 The kanban board on Monday evening (coloured rectangles represent the cards (aka kanbans). Design has finished features 1, 2 and 3 during the day. Coders have taken cards 1 and 2 from the Design "finished" section and pinned them up in coding. Coding work has yet to start on feature 3.

Monday morning, design work starts on the first 3 features, and to represent this development, the designers pin cards for Features 1, 2, and 3 to the Design column. As work on each feature is finished, it is moved to a special area in the bottom part of the board (marked "Finished" in **Fig 1**). Monday goes well. By the end of the day, Features 1-3 have been designed. As cards arrive in the "Finished area" of the design part of the board, the Coders take them and pin them up in the coding area, to show that coding work is now in progress. Meanwhile Design pins up cards for features 4,5, and 6 in the Design column to show what the designers are now working on. The board looks like **Fig. 1.**

Two rules about the "finished" section of the board should be mentioned briefly here, as they are what distinguish a kanban system from a simple visual display. Firstly, notice that each set of people in the production line are responsible for moving the card into their section (we saw that the coders had to move the card from the design-finished area into the coding work in progress area). This ensures that the board represents what people are actually doing, rather than what they are supposed to be doing. Secondly, the project team needs to set rules about the number of kanban that can be in the "finished" area and how long they may stay there. Kanban involves limiting the amount of work in progress, and regarding it as a problem if too much partly-completed work is building up. More on this later.

In this case, Feature 3 was only left in the design-finished section on Monday evening for a trivial reason – the Coders left the office for the day immediately after a meeting and forgot to check the kanban board to see if Feature 3 was ready for coding. That mistake is quickly corrected Tuesday morning.

Before we continue with how this project unfolds, an aside about the practicalities of the "board" itself. While I have mentioned specifics about notice-boards, cards and arranging the board into sections, the important thing, I think is the idea of an easy-to-use status display. The details are probably not so important – one could use sticky notes or pieces of paper instead of index cards, or it would be easy enough to mimic the kanban board in a spreadsheet or table of a word-processing document. Electronic versions have the advantage that they can be shared by teams that do not have an office in which to gather (and no kanbans blow off the board if someone leaves a window open), but they are that little bit harder to keep up to date and maybe more likely to be dismissed as boring project documentation. It is also likely that it falls to one team member to keep the electronic file up-to-date: this undermines the idea that it is each worker's responsibility to report on his or her progress, and makes it more likely that there is a gap between the status board and what is really happening. Similarly, although my example is a software development one, I hope it is easy to see how it could be adapted for other situations – e.g. if a lot of content is being written, the kanban board could be divided into Writing, Editing, Illustrating, Formatting and Approval. Or you could use it in many other situations.

Tuesday – the kanbans advance

Back to the progress of the project. Work continues on Tuesday, with the coders finishing features 1-3 and putting them into the coding-finished area. Testing eagerly snaps up these cards into the testing area. Meanwhile Features 4-6 have reached coding and design working on Features 7-9. These developments are shown on the kanban board by moving the cards along the board, so that by Tuesday evening it looks like **Fig 2**.

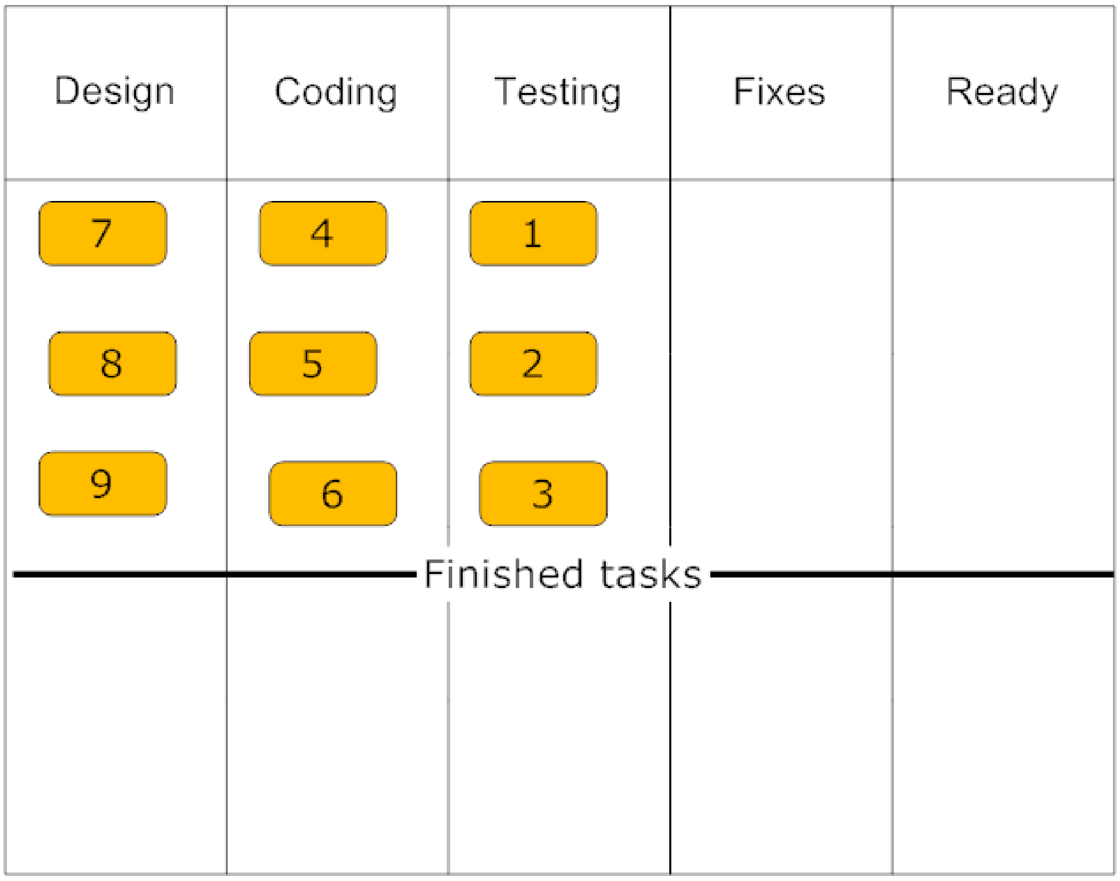

Figure 2 The kanban board on Tuesday evening

Wednesday – work in progress limits

So far, the kanban board has only really functioned as a useful visual status report – each worker can see what everyone is working on, and the flow of cards across the board shows us all how it is going, and what is going to be hitting our part of the production line soon. It is not until Wednesday, we hit our first problem, that we see some other features of kanban. On Wednesday evening, the kanban board has reached the interesting state shown in **Fig 3** (overleaf). Feature 5 is taking longer to code than we thought. Coding of Features 4 and 6 finished on schedule, and the designers kept up to schedule too, designing features 10- 11. On the plus side, Features 1 and 3 sailed through testing without any problems, and only Feature 2 needs some fixes.

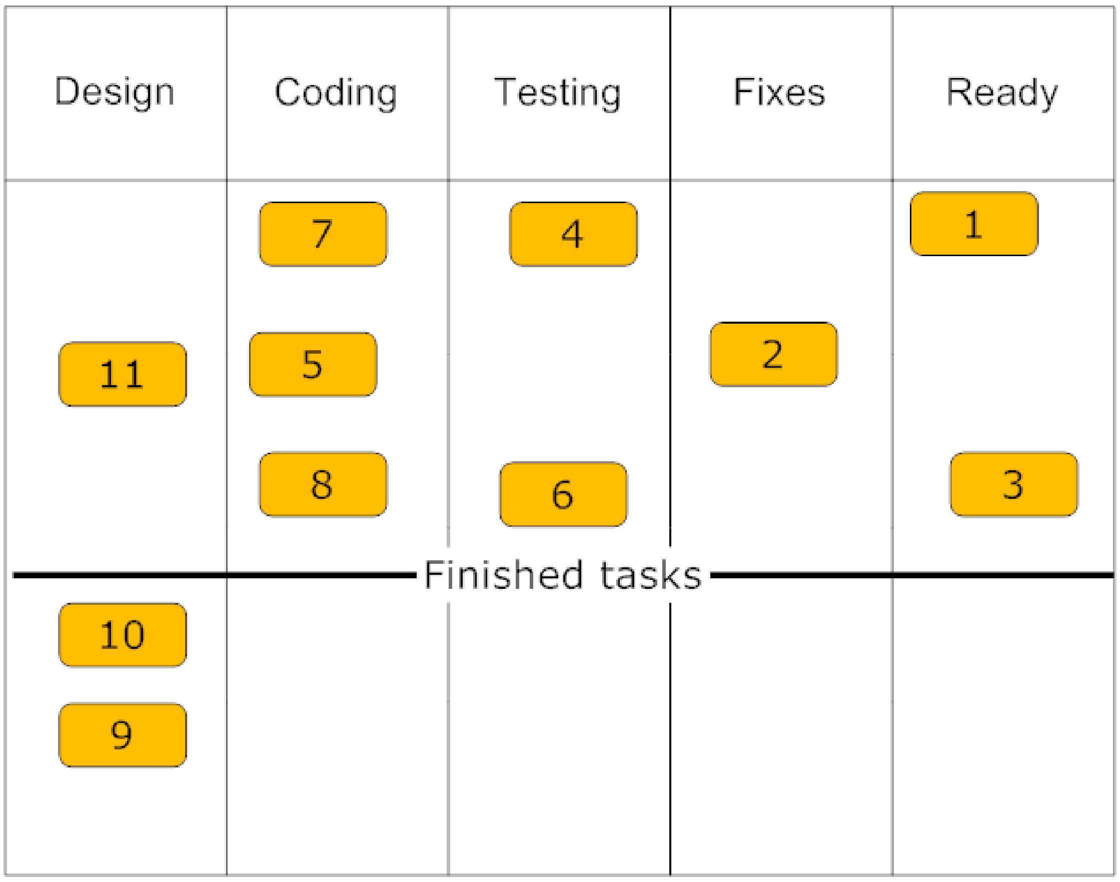

Figure 3 kanban board on Wednesday evening. Feature 5 is behind schedule, preventing coding from taking up features 9 and 10

Coding is already fully busy with Features 7, 8 and the troublesome Feature 5. So coding cannot take Features 9 and 10 from the design-finished section of the board. They are stuck in design-finished "for real" rather than for trivial reasons such as someone not checking and updating the board. As mentioned earlier, an important feature of kanban is that it should include features to limit the amount of work in progress. In this case, lets assume that either the number of features in "Finished", or the amount of time they've been there exceeds what the team agreed. In some project management practices a nice, full in-tray of work might be seen as a good way to keep coding focused. In a kanban system it is seen as a problem that the team needs to rectify urgently.

On Thursday morning, the team reviews the kanban board and can quickly see the problem, and start discussing what to do. Thursday looks like a quiet day for Testing and Fixing (2 and 1 features, respectively, to work on). Perhaps team members from there can help out either finishing Feature 5 or starting on Feature 9. Or perhaps there is a problem with coding the designs that Design is producing, and this needs to be understood lest the problem repeat itself with further features.

You can see how this would go on, I think, so I will not describe Thursday, Friday and further days.

Limiting work in progress

When I first began reading about kanban, I found the idea of limiting work in progress quite counter-intuitive. Assuming that the folks doing the upstream processes to the blockage cannot help out with overcoming the blockage, a lot of project managers would, I think, decide "let's at least make the progress we can make" and keep the upstream folks going and adding to the backlog. Kanban, by contrast, focuses on achieving a smooth flow through the process and (ironically) this may mean a lot of stops and starts to begin with, as to many things pile up in "finished" and the team stop to sort the process out (and gets a better idea of the appropriate rules for the number of items that may be in "finished, and how long they should be allowed to stay there).

This methodology makes a lot more sense when one considers that having a lot of work in progress represents a risk to the project. The risk is one of irrelevance – effort is wasted on doing work that cannot be used in the final project. I gave one possible case in the example – when coding cannot complete Feature 5, this could be because of some mismatch between what design are requesting and what coding can do. If that is so, then design need that feedback quickly: either the features need to be rethought to be more code-able, or the instructions clearer, or there needs to be re-estimation of how long coding is going to take (and whether anything ought to be done about this). Another possible risk is that work is done on the early stages of features that then need to be changed significantly or abandoned and the project proceeds. I suppose that we are comparatively lucky in software – in manufacturing or construction, work in progress can cause other problems (e.g. where to store it all, or that it might be perishable).