Using **Kanban** Techniques to Control Incremental Development

Jeff Patton
AgileProductDesign.com
jpatton@acm.org

Download this presentation at: www.agileproductdesign.com/downloads/patton_kanban.ppt

In this short talk we'll cover:

1. What is a Kanban System and how does it apply to software development?
2. How to set up a development team Kanban System
3. Applying Lean thinking to software development

看板 – Kanban cards limit excess work in progress

看板 – Kanban literally means “visual card,” “signboard,” or “billboard.”

Toyota originally used Kanban cards to limit the amount of inventory tied up in “work in progress” on a manufacturing floor

Not only is excess inventory waste, time spent producing it is time that could be expended elsewhere

Kanban cards act as a form of “currency” representing how WIP is allowed in a system.

Kanban simulation

Let's simulate a simple process, then see if we can improve it by adding a Kanban system.

I'll need 5 volunteers to manufacture the latest in high-tech aircraft

Why use Kanban in Software Development?

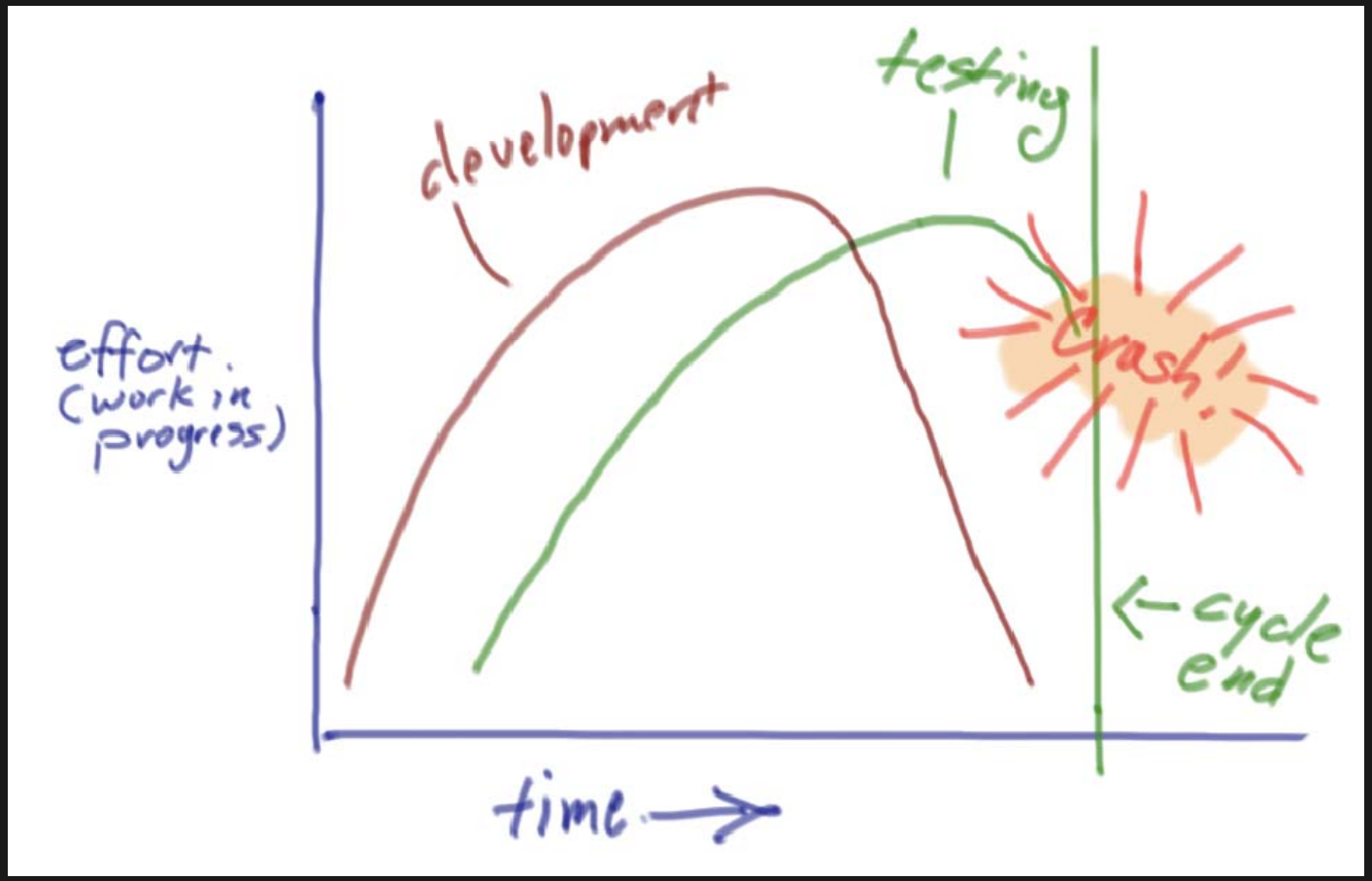
(we're not building aircraft – or anything tangible really)

Time-boxed iterative development has challenges

Common problems include:

- Short time-boxes give more frequent opportunity to measure progress and inspect software but force development items to be smaller
- Smaller development items are often too small to be valuable and difficult to identify
- Quality of requirements suffers as analysts rush to prepare for upcoming cycles
- Quality of current development suffers when busy analysts are unable to inspect software or answer questions during development
- Quality often suffers as testers race to complete work late in the development time-box

Inside an iteration, effort across roles is uneven

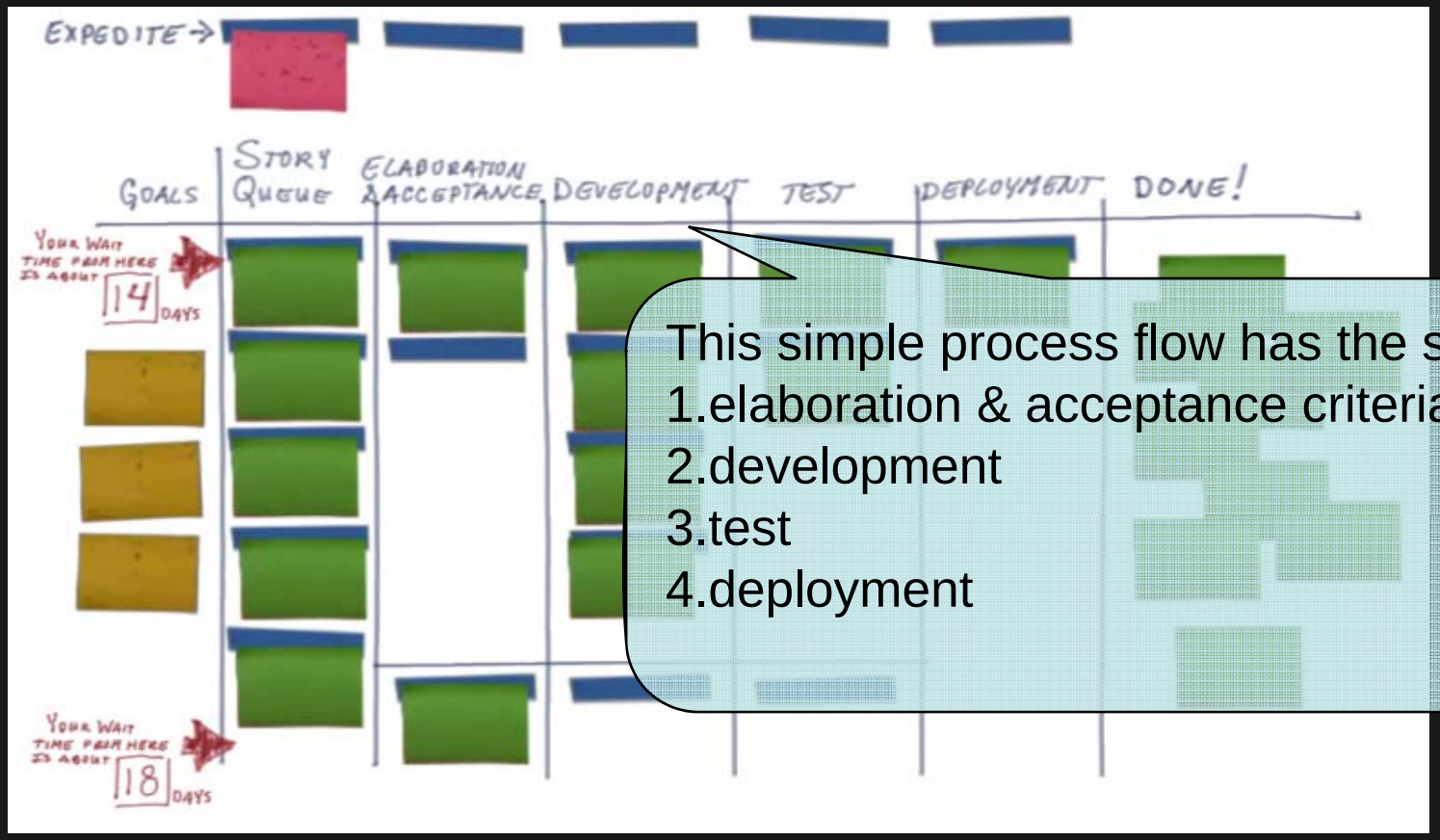


Development work often continues throughout a cycle while testing starts late and never seems to get enough time

Using a Kanban approach in software drops time-boxed iterations in favor of focusing on continuous flow.

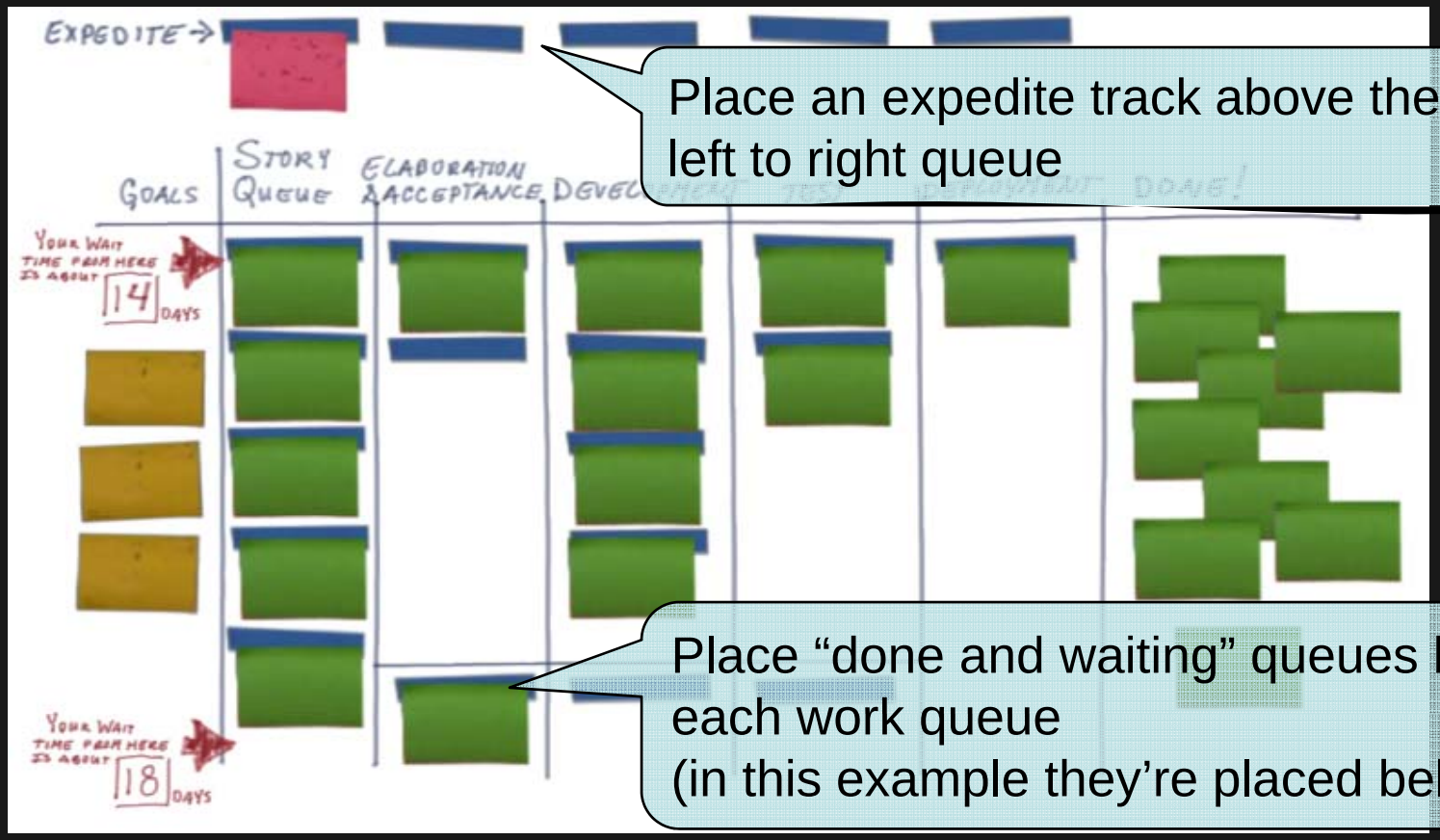
How to set up a simple Kanban system for a software development team.

1. Define a work process flow



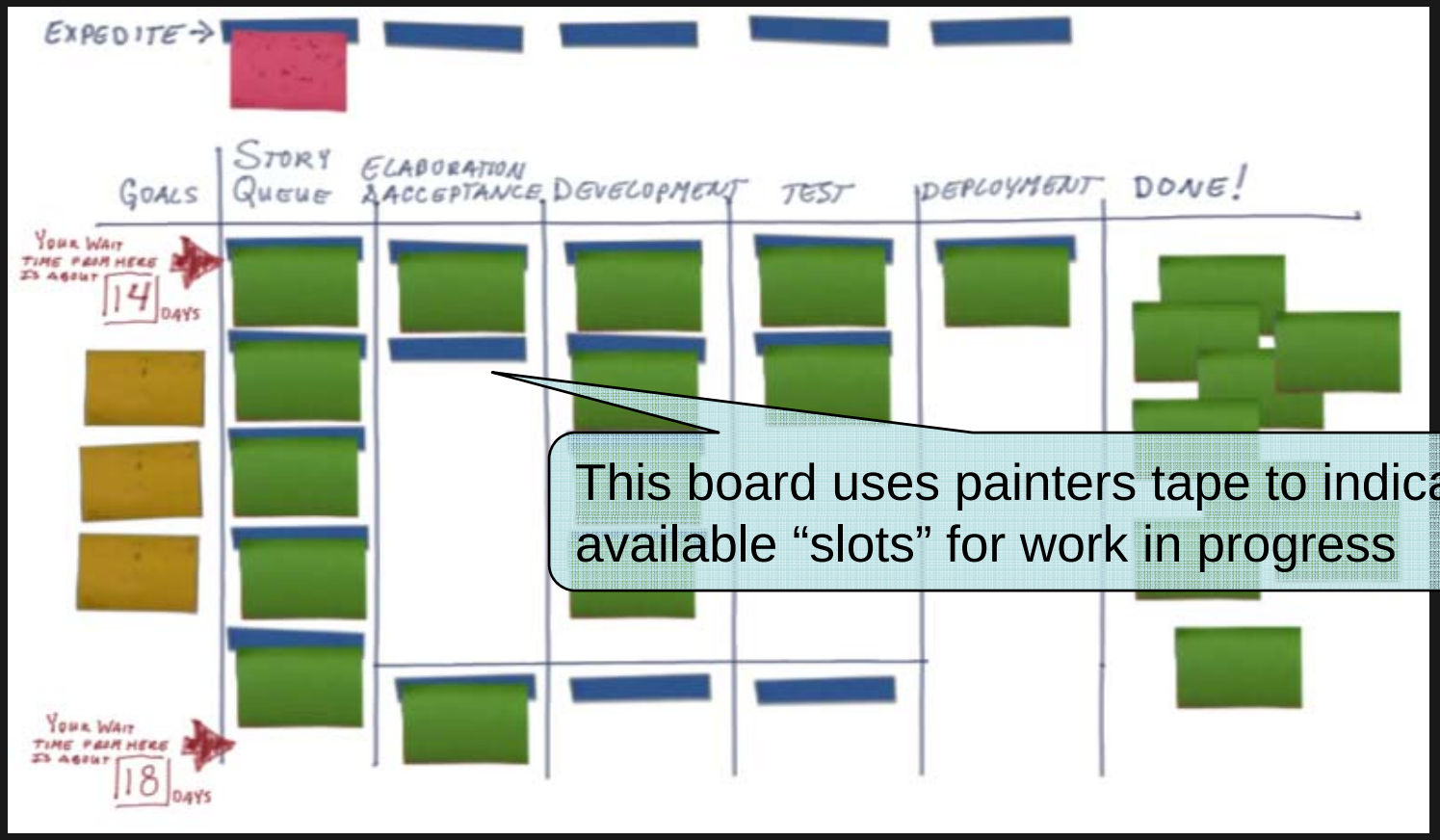
Look at the typical flow for features, stories, or work packages and describe typical process steps

2. Lay out a visual Kanban board



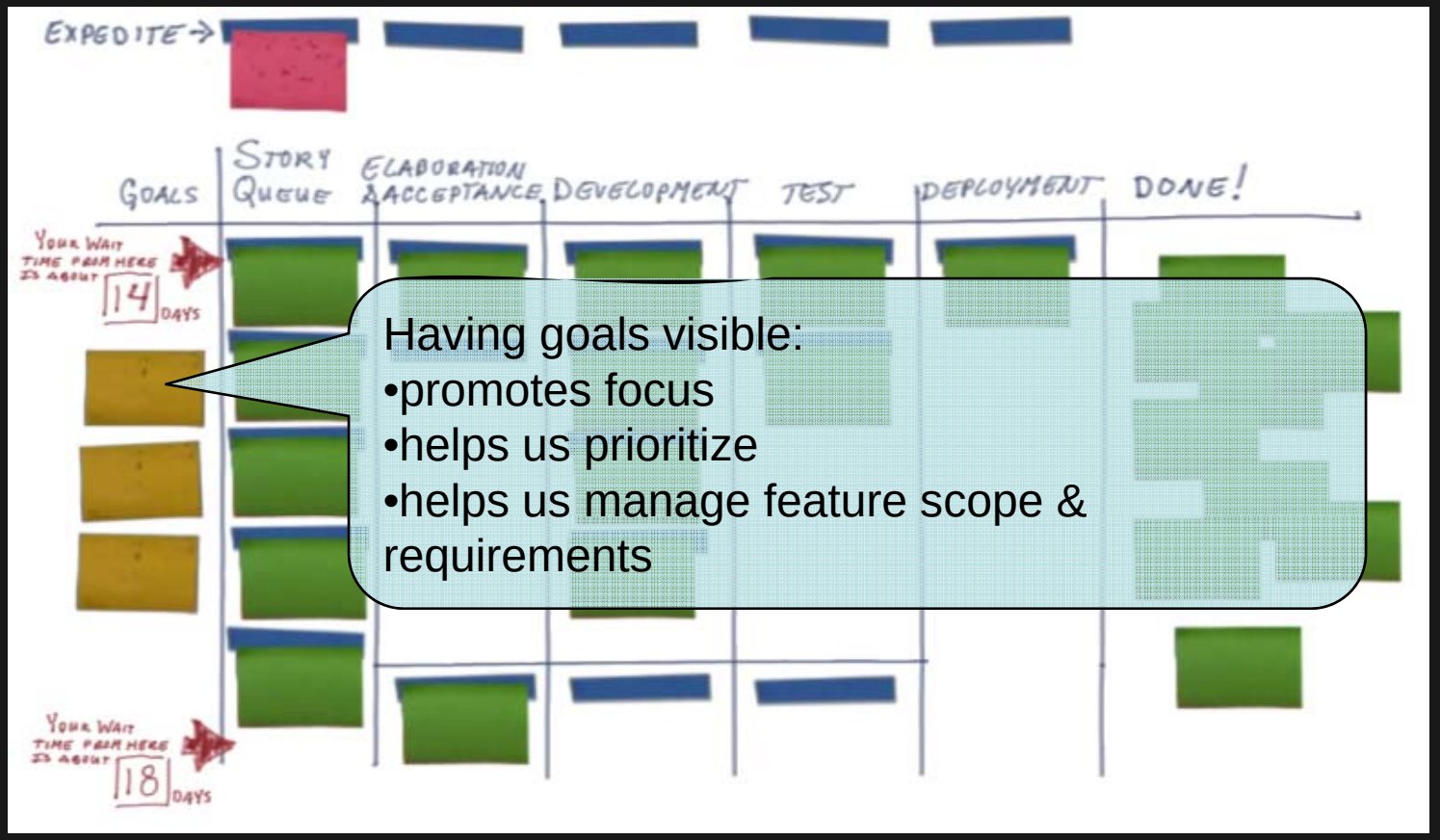
Place a goals column on the left, then a waiting queue, the process steps, and a final “done” column to the right

3. Decide on limits for items in queue and work in progress



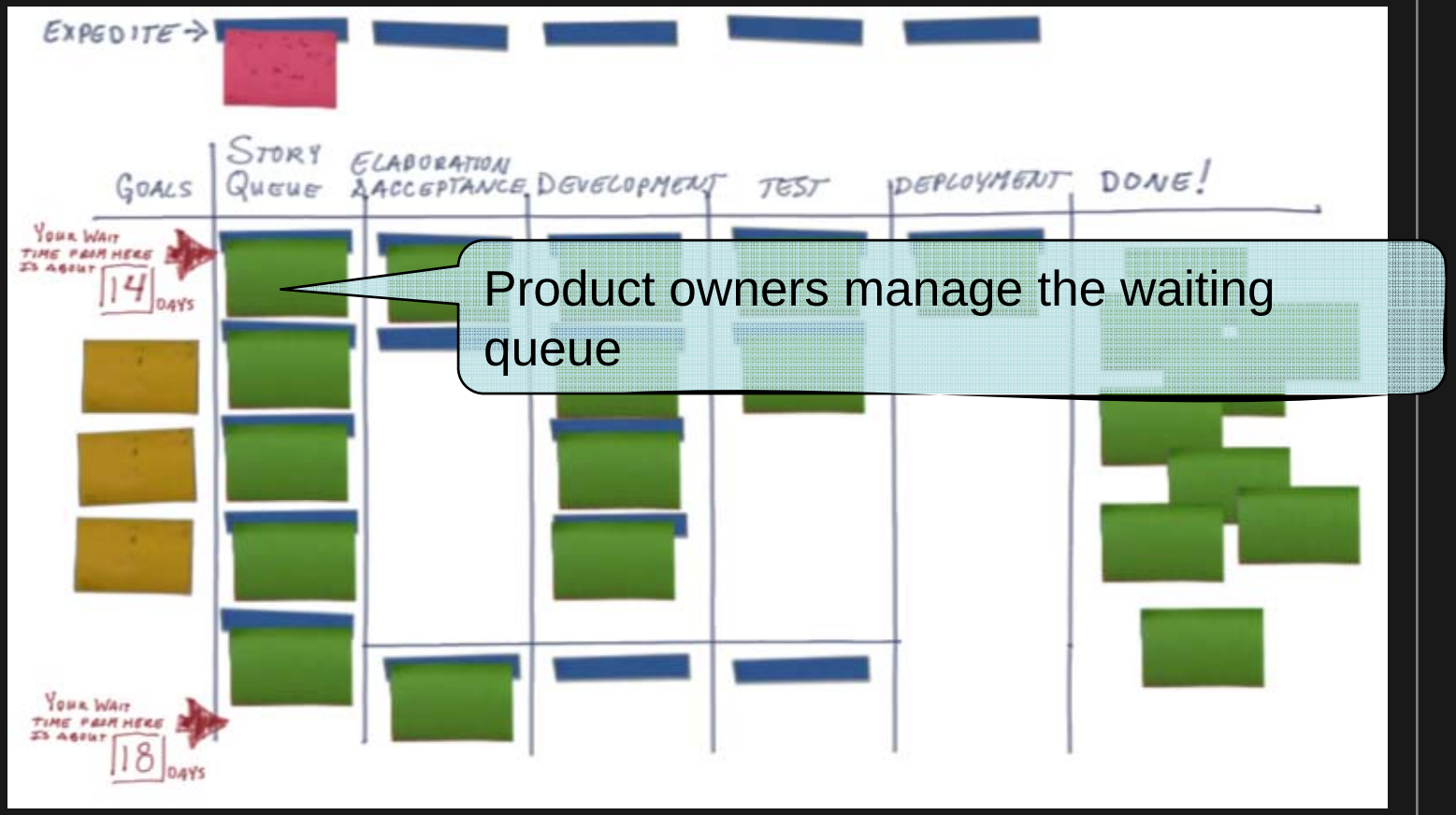
A good limit is a factor of the number of people in a role that can work on an item in a given process step. Start with number of people * 1.5

4. Place prioritized goals on the left column of the board



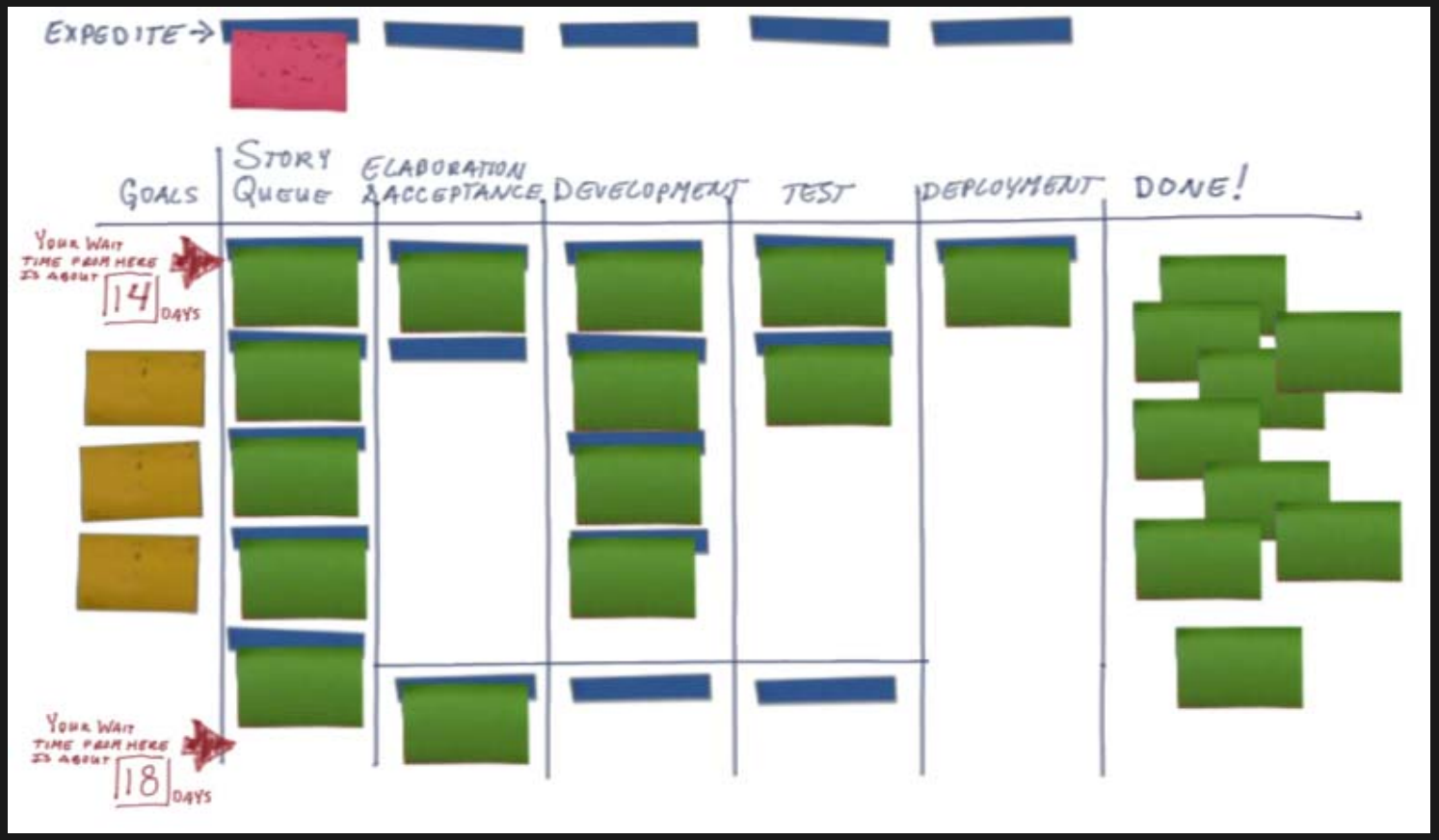
A good goal describes the outcome we hope to achieve after software ships. Goals help keep focus on the larger outcome.

5. Start the board by placing stories or features in queue



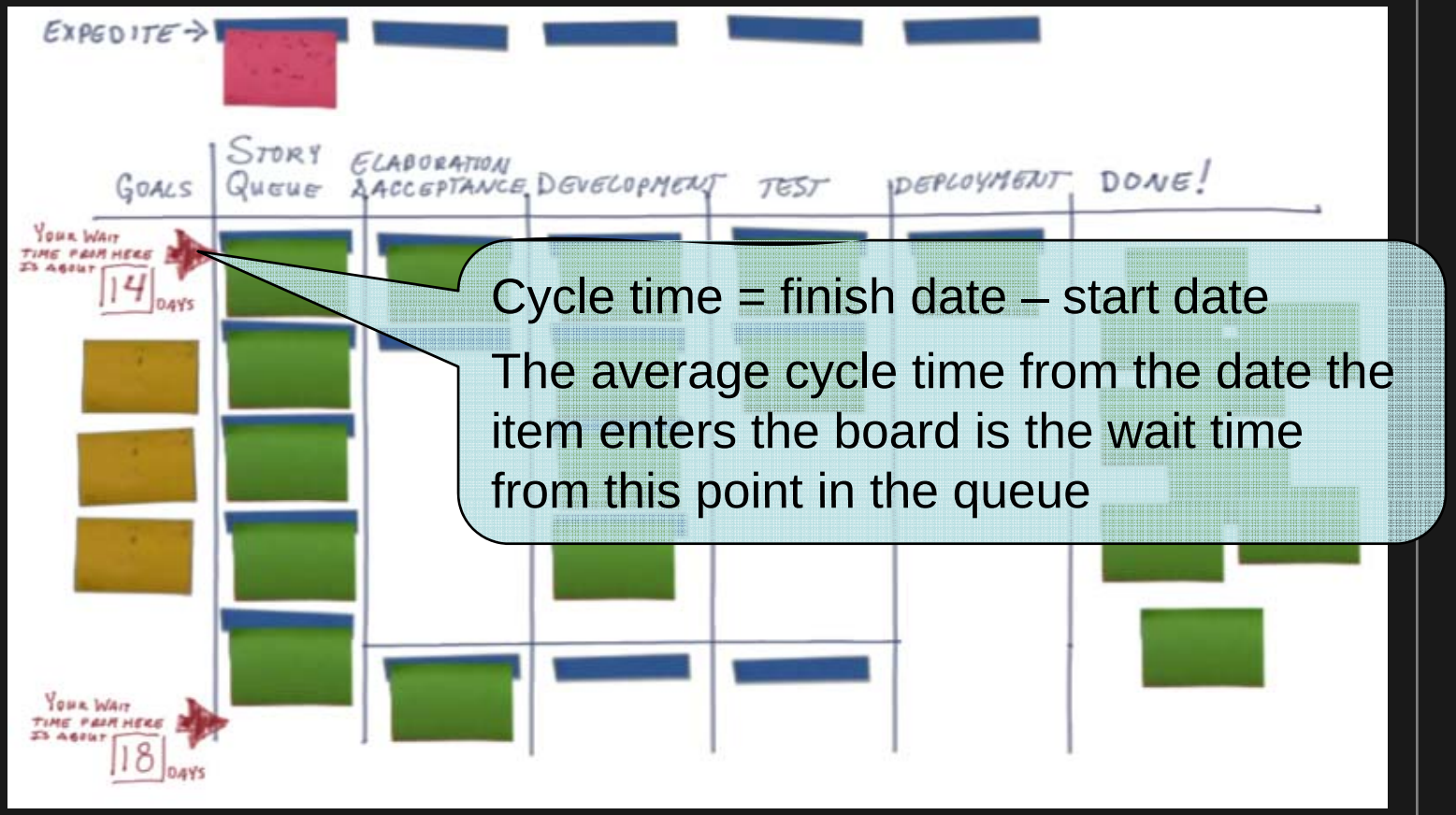
Mark on the story or feature card the date it entered the queue. This begins our measurement of cycle time.

6. Move features through the process flow as work is completed



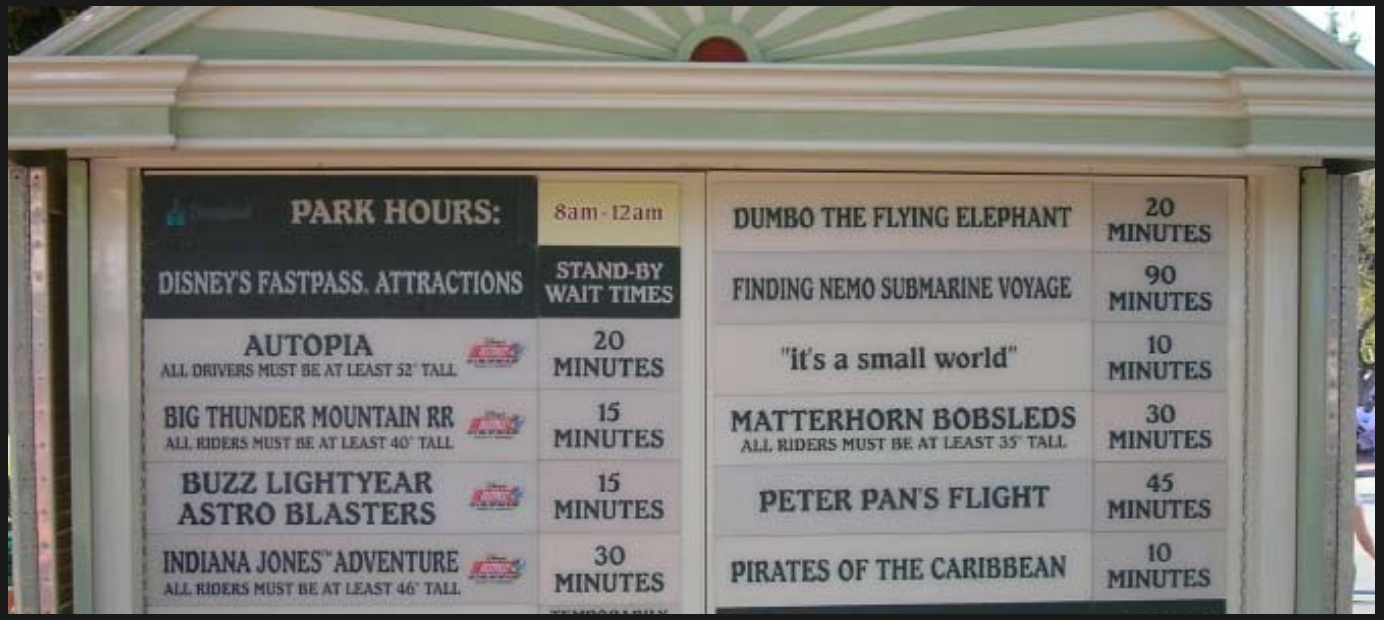
As the story enters the first process step, mark that date on the card. This is the start date. As it's finished, mark that date on the card. This is the finish date.

7. Use the dates on the cards to calculate cycle time



Use average cycle time to set wait times from different points on the board. Pay attention to flow and bottlenecks: relieving bottlenecks as quickly as possible.

Display and manage cycle times



Disneyland's public display of cycle-times

Reduce the number of Kanban slots allowed until cycle time remains unchanged

Reduce the size of development items

•Work in progress is actually the number of items * the average size of items

Identify and act on bottlenecks immediately

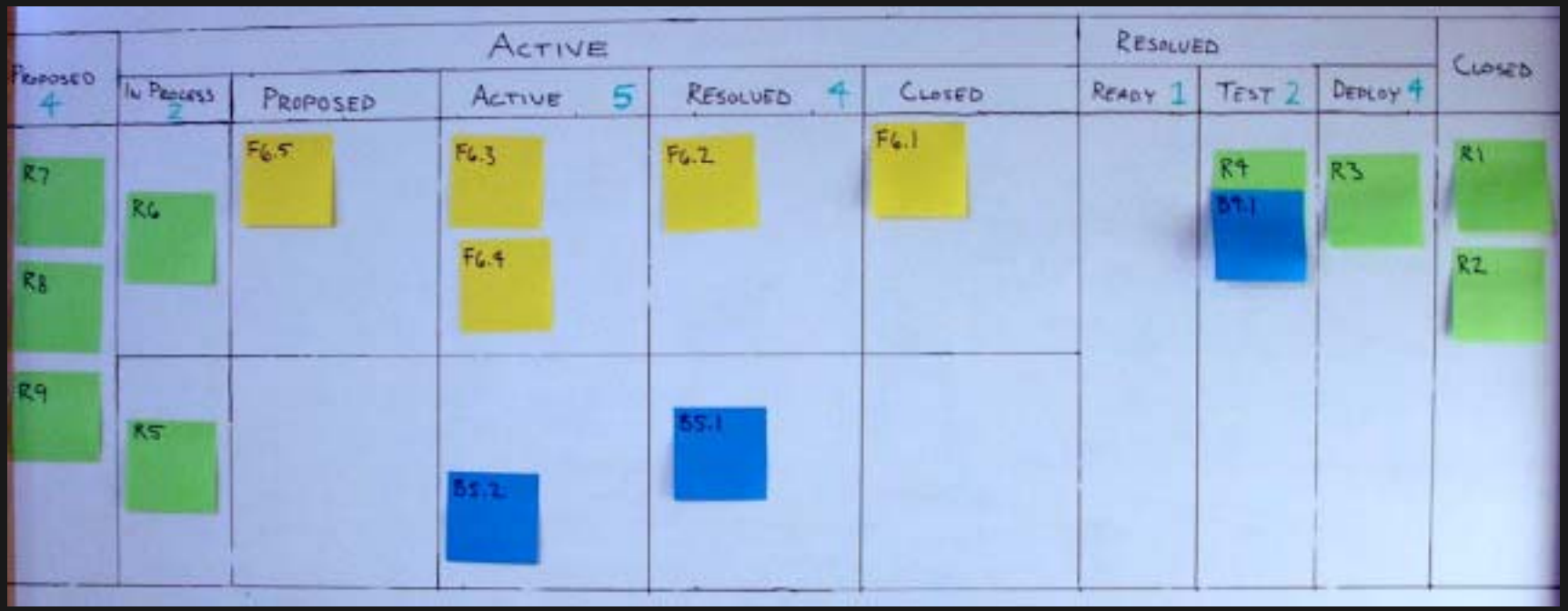
•Relieve repeated bottlenecks by changing the number and types of people in each role and cross training

Kanban Boards

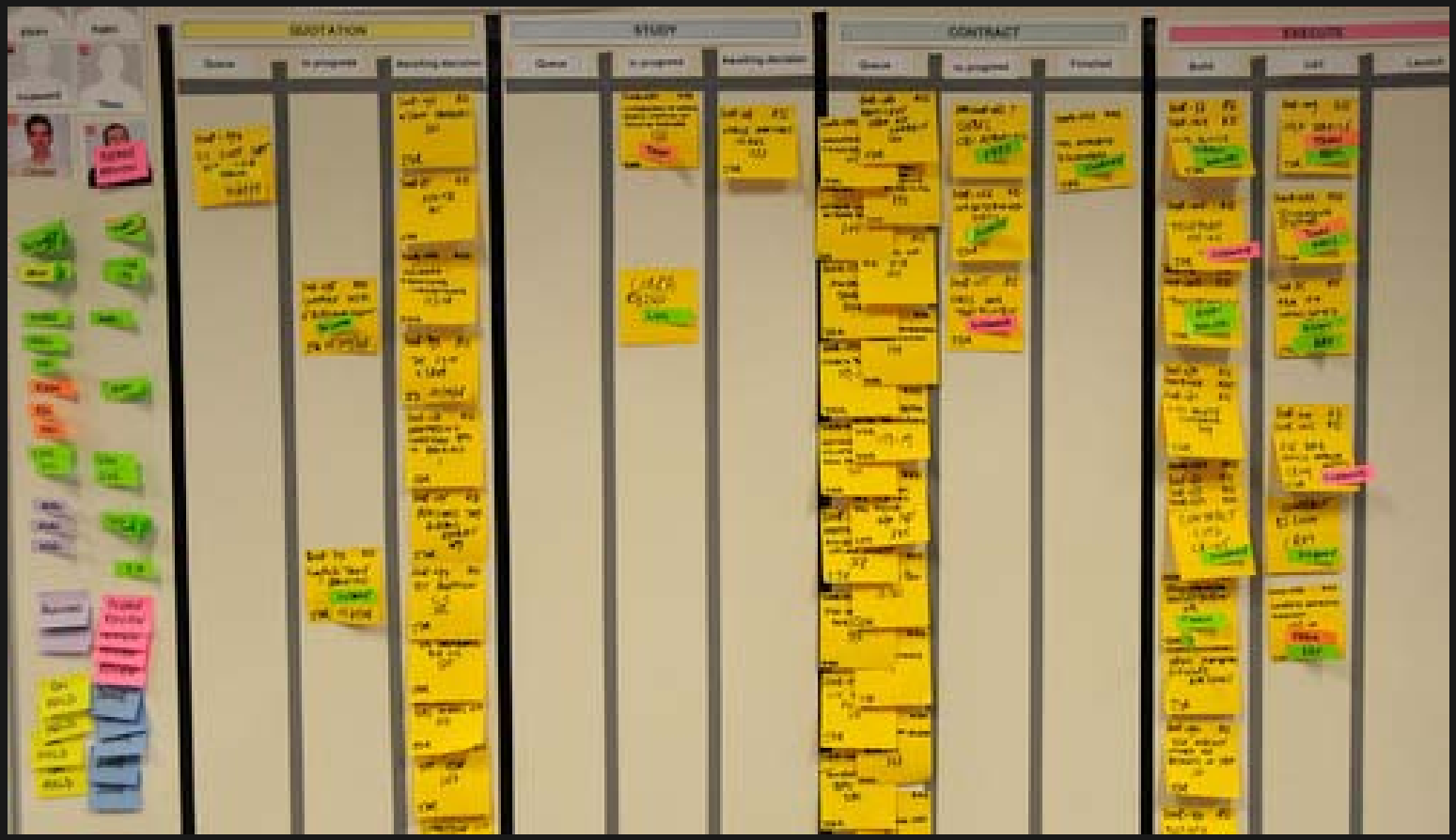
Kanban Boards

Kanban Boards

Kanban Boards



Kanban Boards

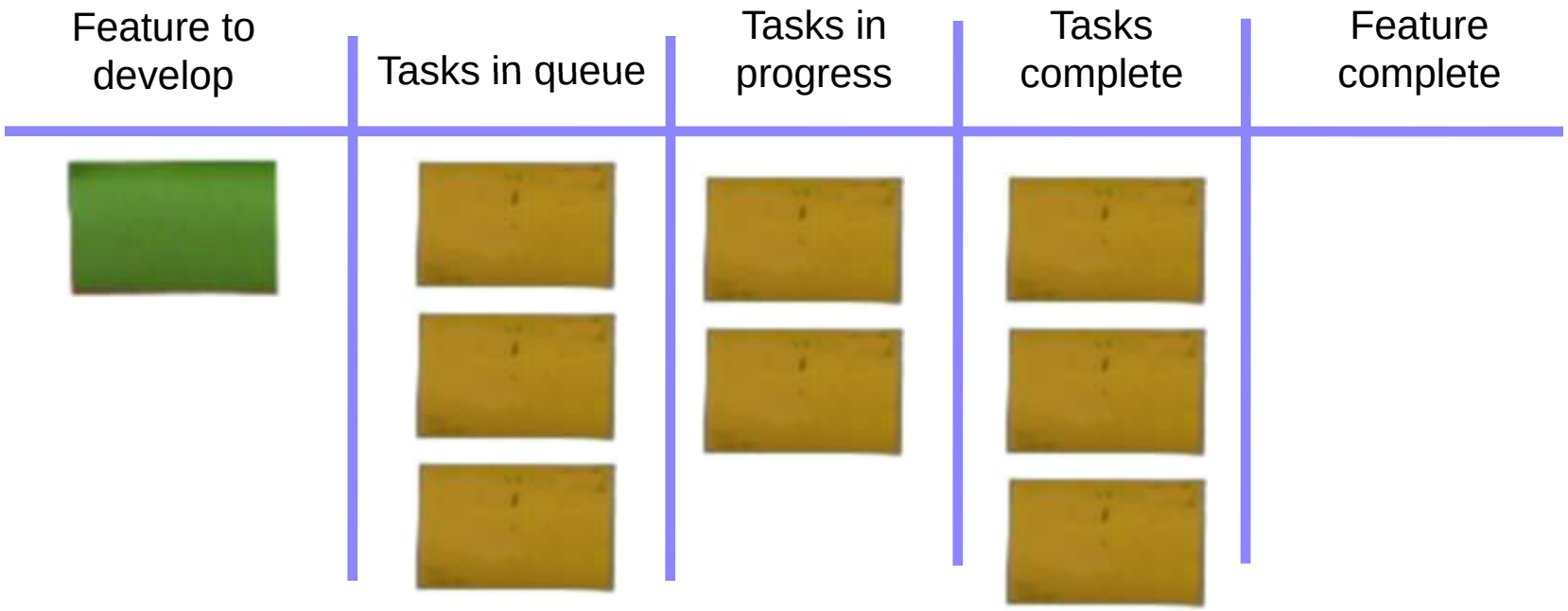


Explode large process steps into tasks to improve visibility

When a feature, user story, or work item is large:

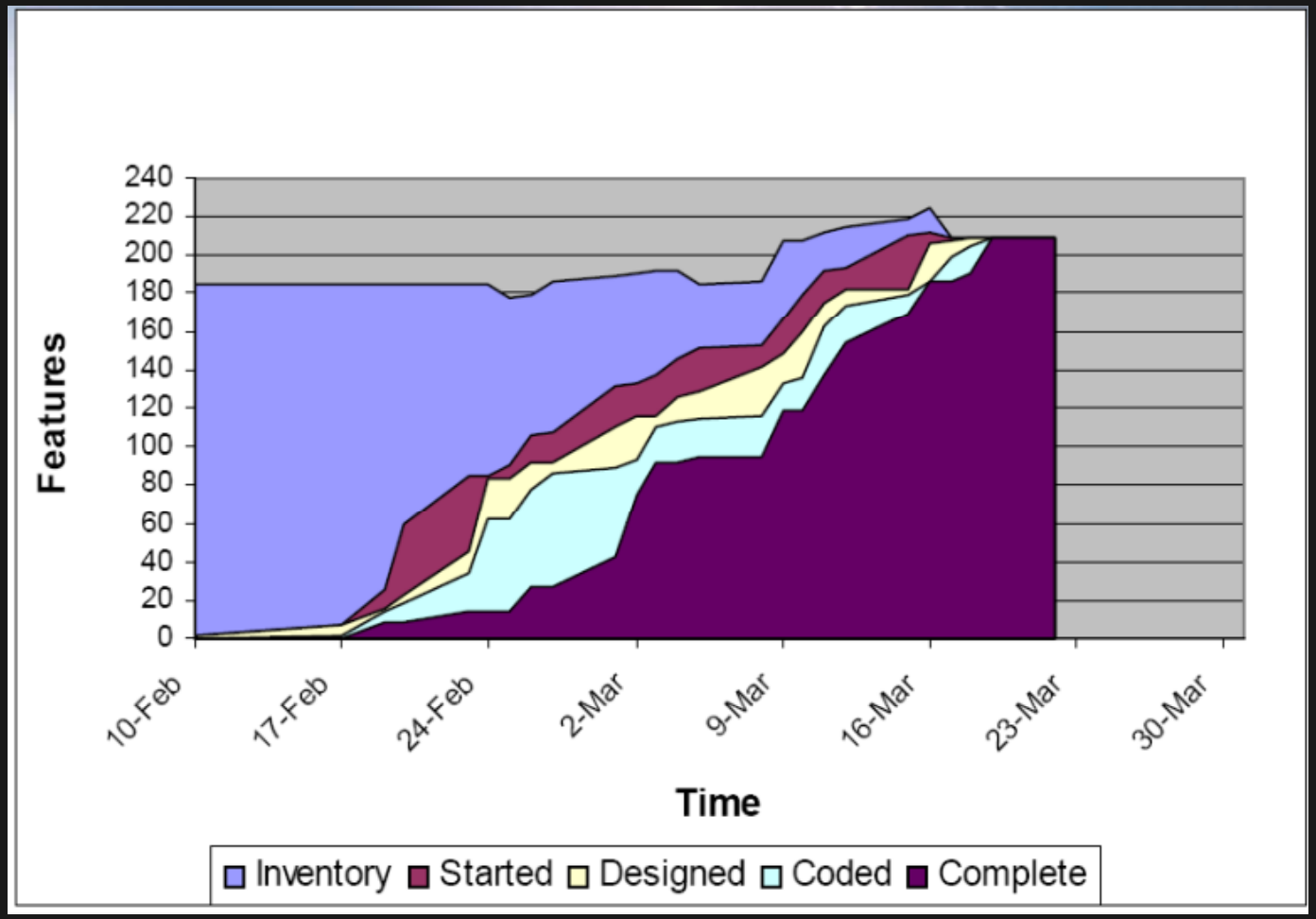
- Takes longer than a couple days to complete
- Requires that multiple people collaborate on its completion

Decompose that step into cards to track independently



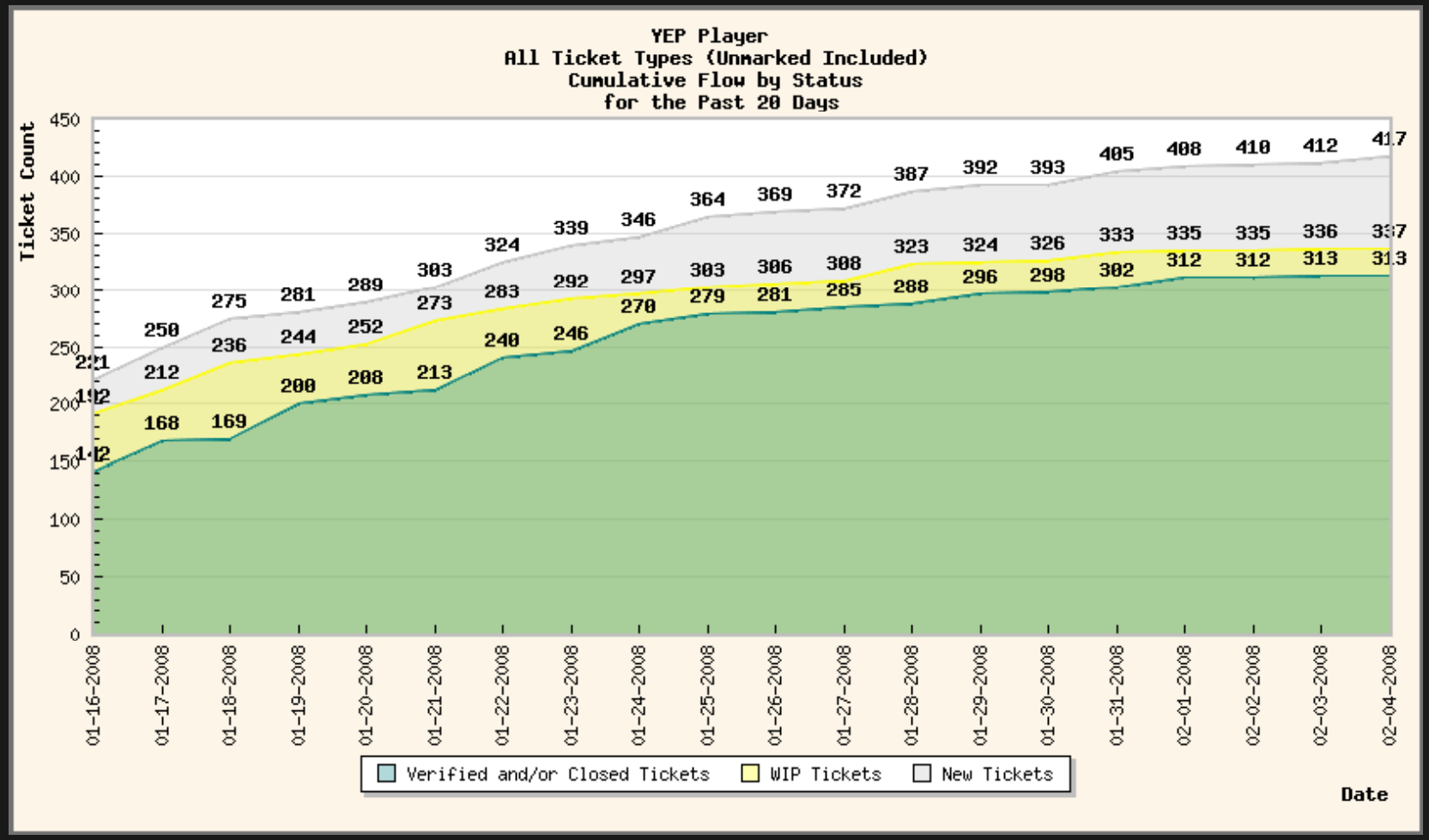
Kanban Board with Task Decomposition

Use cumulative flow diagrams to visualize work in progress



www.agilemanagement.net/Articles/Papers/BorConManagingwithCumulat.html

Use cumulative flow diagrams to visualize work in progress



www.agilemanagement.net/Articles/Papers/BorConManagingwithCumulat.html

Keep time-boxed product and process inspection

Keep regular time-boxes in your process as a cue for **product** inspection:

•Evaluate the quality of the growing product from a functional, engineering, and user experience perspective

Evaluate your **pace** of development:

•Look at the number of development items completed relative to goals

•Look at the average cycle time per development item

•Calculate the ratio of developer days per completed item. Use this ratio to estimate the completion time for undeveloped items

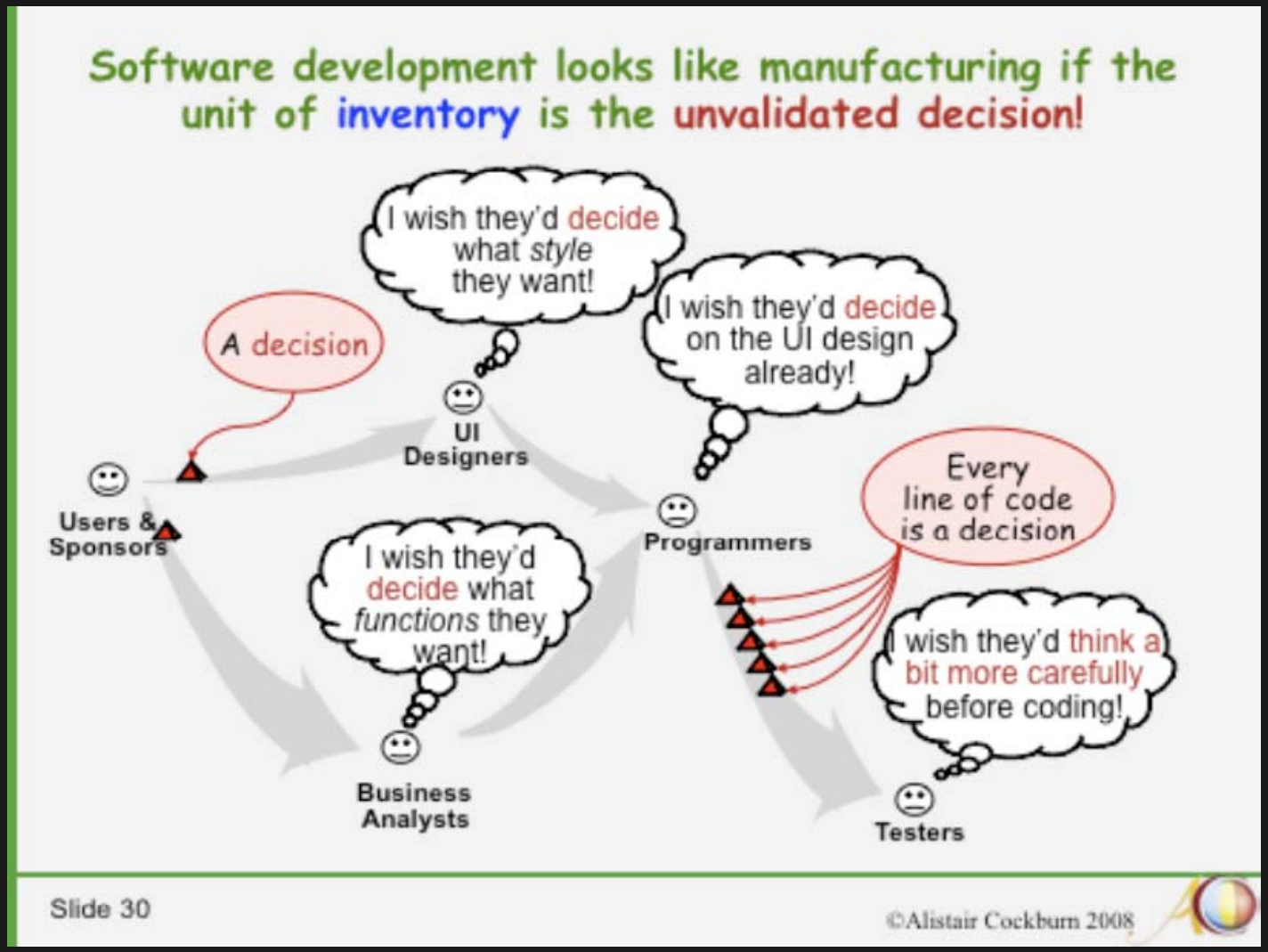
•Adjust your development plan as necessary

Evaluate and adjust the **process** you're using

•Use a process reflection session to identify changes you could make to improve your product or pace

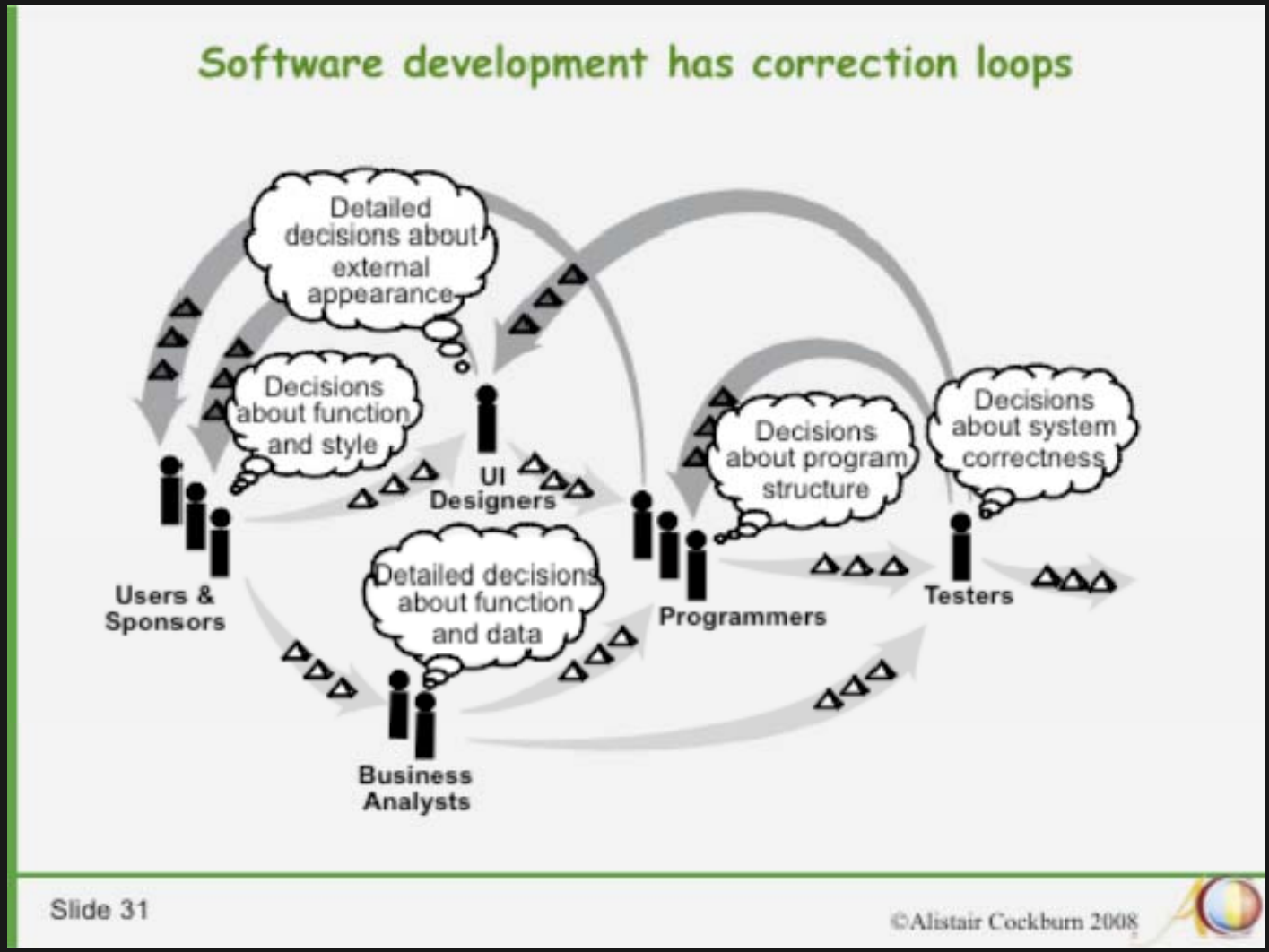
Ending cycles right: http://www.stickyminds.com/s.asp?F=S14865_COL_2

Begin looking at your process using Lean thinking



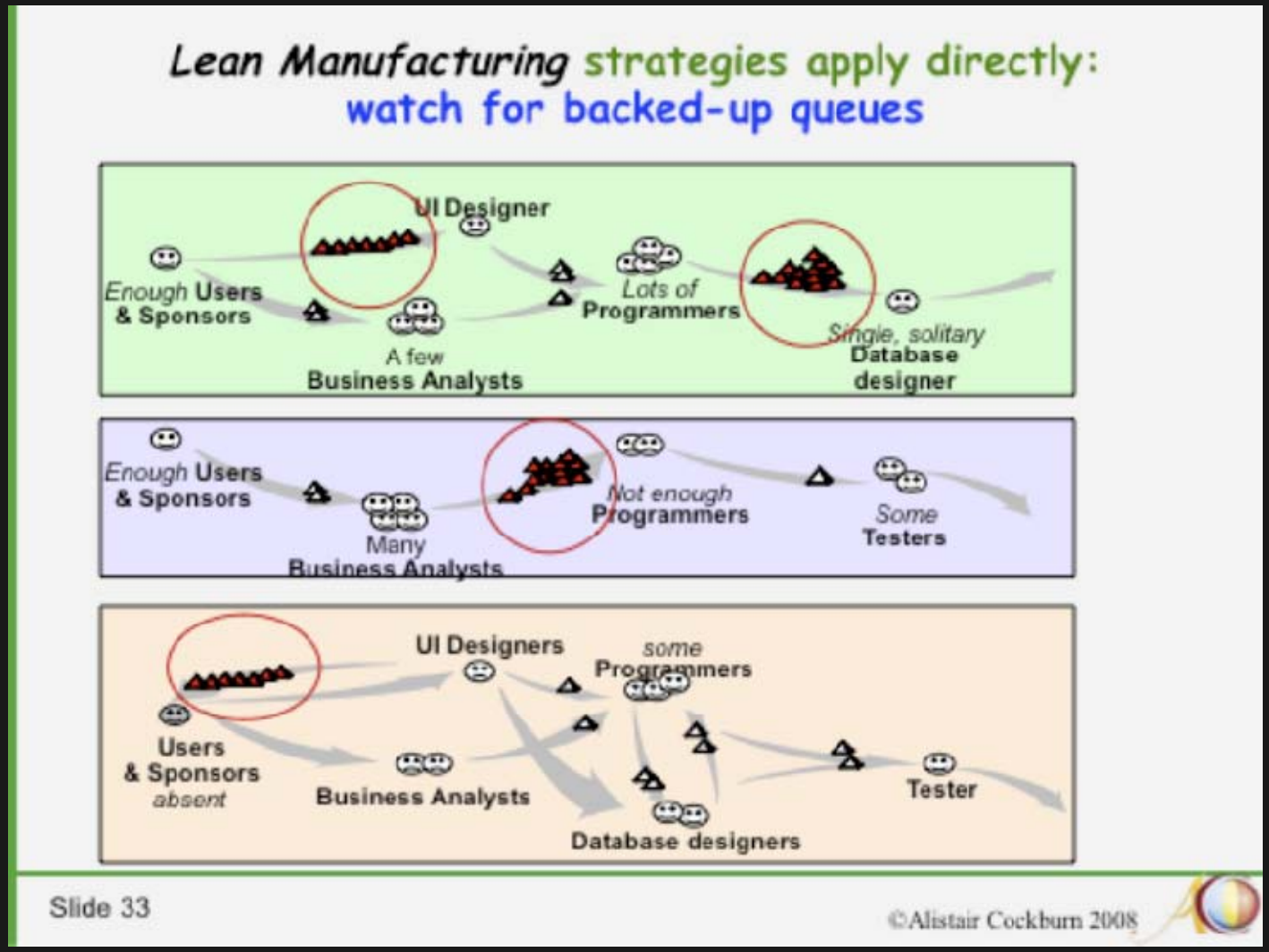
Cockburn's Software Engineering in the 21st Century:
http://alistair.cockburn.us/Software+engineering+in+the+21st+century.ppt

Since we're engaged in "knowledge work" look at the cycle time of validated decisions, or knowledge



Cockburn's Software Engineering in the 21st Century:
http://alistair.cockburn.us/Software+engineering+in+the+21st+century.ppt

Often the feedback loop is overlooked – it's the invisible backed-up queue



Cockburn's Software Engineering in the 21st Century:
http://alistair.cockburn.us/Software+engineering+in+the+21st+century.ppt

Setting up a simple Kanban system starts to focus the team on the cycle-time of delivered work and gives a way to detect and begin to resolve bottlenecks

Kanban References:

- Anderson, **Kanban in Action**: http://www.agilemanagement.net/Articles/Weblog/KanbaninAction.html
- Hiranabe, **Kanban Applied to Software Development: from Agile to Lean**: http://www.infoq.com/articles/hiranabe-lean-agile-kanban
- Ladas, **Scrumban - Essays on Kanban Systems for Lean Software Development**: http://www.lulu.com/content/3864767
- Ladas, **Scrum-ban**: http://leansoftwareengineering.com/ksse/scrum-ban/
- Belshee, **Naked Planning, Kanban Simplified**: http://joearnold.com/2008/03/naked-planning-kanban-simplified/