Agile for Large Systems

Agile Meets Lean: Methods for Managing Large Projects

By Kathy Iberle

Key Points

- **Agile works well because it limits both *batch size* and *work-in-process*, and improves *flow*.**
- **The methods Agile uses to accomplish this were designed for small to medium organizations.**
- **Lean provides more sophisticated methods to reach the same goals in larger organizations.**

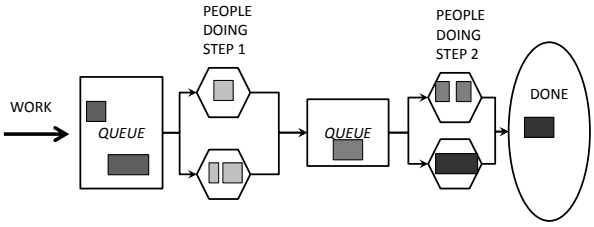
What Makes Agile Work?

Agile methods work because they do a very good job of optimizing the flow of work through a team. They deliver value quickly and cut waste in numerous ways. The same type of benefits can be obtained in multi-team development systems. However, first you'll need to explicitly understand *how* agile methods control the workflow.

We'll start by envisioning the development process as a system or machine turning ideas into saleable stuff. The "stuff" is divided into a set of discrete chunks or ***batches***, each of which delivers value. Each batch moves through a series of activities. When an activity finishes with a batch, the batch goes into a waiting area or *queue* to wait for the next activity.

Agile does several things which increase the output of batches:

- Split the work into small batches which deliver value.
- Limit the number of batches under development at a given time.
- Manage the batches on a cadence.
- Make the progress of the batches visible.
- Optimize progress of batches through the system.



Small batches are good because they provide much faster feedback than large batches. Rapid feedback is especially valuable when the business demands change rapidly or there are a lot of technical unknowns. We look for:

1. Feedback from the sale, review, or actual use of the finished batch – are we making the right thing?
2. Feedback between steps of the development process – are we making the thing right?
3. Feedback on the overall progress of numerous batches – how long will it take to finish all the batches?

A batch must ***deliver value*** to a customer, so that increasing our output of batches will increase the stuff we have to sell or use. We don't want to simply increase the output of work done. Most of us don't need more work!

Limiting the number of batches which are active at any given time cuts down the time spent keeping track of the ongoing batches. More importantly, it slashes task-switching time - the time people spend switching from one job to another during the day. Task-switching is a huge but invisible drain on people's time in most knowledge work.

The mechanisms which manage the flow of batches are put on a ***cadence***, so coordination activities such as meetings and decisions occur on a predictable, frequent rhythm. This predictability reduces the time needed to manage them.

The progress of batches through the system is ***made visible***. Once the progress of each batch is visible, less time is spent on gathering status, and bottlenecks and inefficiencies can be more easily seen.

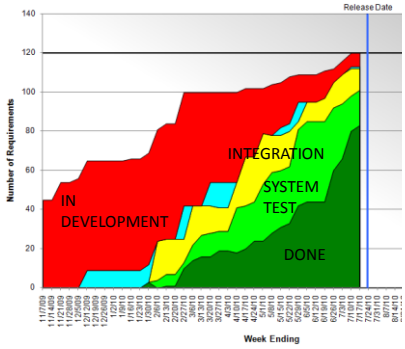
Agile for Software

Scrum, a popular agile method, applies these concepts to software development.

- The ***batch*** with ***value*** is defined as a user story, which is a piece of functionality that lets a user do something.
- The ***number of batches in progress*** (known as *work-in-process*) is controlled by setting up time boxes known as sprints. Each sprint accepts only as many user stories as it can realistically finish in a sprint.
- ***Cadence*** is applied by making all the sprints the same length, so meetings and decisions are on predictable dates.
- Progress is ***made visible*** daily via a *Visual Planning Board* and stand-up meetings.

In a large organization, work flows *between* teams rather than inside a single team. The same concepts apply to larger systems, but the appropriate methods aren't necessarily the same.

- ***Split the work into small batches***, each of which delivers value to your customers. The batches have to be much smaller than an eighteen-month project, but they don't have to be a two-day user story.
- ***Make the system visible*** with a process diagram *or value-stream map*. People in a large, complex system often can't accurately see how teams are interacting. Simply mapping the flow may make things better!
- ***Make the progress of batches through the system visible.*** A Visual Planning Board is often used for daily snapshots, accompanied by a *cumulative-flow diagram* (right) to show progress over time.



- Now that you can see what is happening, ***optimize the flow of batches*** through your unique system. This often involves:
    a. Simplifying or clarifying handoffs and feedback loops.
    b. Changing the batch size.
    c. ***Limiting the number of batches in progress*** at any given time to reduce task-switching.
    d. ***Instituting cadence to reduce overhead***. Large systems may need two (or even three) nested cadences.

Once you know why agile works, it's possible to get the same benefits on larger projects. Surprisingly, it ***is*** possible to reduce time to market and increase your delivery of value without hiring more people or expecting more hours. It's all in how the workflow is managed.

Learn More

Curious about Lean methods? Visit my website www.kiberle.com to see more booklets like this one, as well as longer papers and other resources. Or give me a call!

IBERLE CONSULTING GROUP, INC.

13504 NE 84th St.
Suite 103-241
Vancouver, WA 98682
360.852.1676
www.kiberle.com
kiberle@kiberle.com