PROJECT LIFECYCLES

How to Reduce Risks, Release Successful Products, and Increase Agility

JOHANNA ROTHMAN

Author of *Create Your Successful Agile Project*

Project Lifecycles

How to Reduce Risks, Release Successful Products, and Increase Agility

Johanna Rothman

Project Lifecycles

How to Reduce Risks, Release Successful Products, and Increase Agility

Johanna Rothman

Practical ink

Ebook: 978-1-943487-31-8; Paper: 978-1-943487-32-5; Hardcover: 978-1-943487-33-2

Chapter 7. Agile Approaches

Agility requires culture changes to manage the team, product, and management needs and risks. (See Section 1.1: Visualize a Successful Agile Team Culture on page 2 for more details. But an agile culture is not sufficient.

While each agile team is unique, each agile team has similar characteristics.

7.1. Characteristics of an Agile Team

All agile teams, regardless of their approach, have these characteristics in common:

- The team limits its WIP.
- The team defaults to collaboration over solo work.
- Regularly, the team delivers value, at least internally, if not to an external customer.
- The team realizes that while features might be done for now, the requirements, architecture, and user experience are not done until this version of the product ships.
- The team retrospects and considers what to do for continuous improvement.

When teams work like this, they work in flow efficiency, controlling their work themselves.

Because they release and retrospect, they have many fewer unplanned feedback loops. These characteristics allow the team to manage its project, product, and portfolio risks.

That's it.

Teams choose practices that allow them to create their specific agile culture. One team might use iterations (a short timebox) to manage its WIP over a one- to two-week period. A different team might create WIP limits for the overall team.

Some teams do both because they learned that just limiting overall WIP doesn't work because they have a tendency for work to accumulate in one area. (That accumulation tends to occur where the team does not have sufficient skills and capabilities to finish the work.)

However, when teams choose to collaborate on a given feature, the team delivers value as early as possible. That delivery allows the product leader to assess the feedback inside and outside the organization, selecting the next bit of work.

The earlier the team can deliver the value, the fewer unplanned feedback loops anyone encounters. That value allows the team to recognize that feedback might cause changes to the requirements, architecture, and user experience.

That is, the team recognizes the requirements, architecture, and user experience cannot be "complete" until the product ships.

Why Specify User Experience as Incomplete?

You might agree with me that an agile approach acknowledges the requirements and architecture will change. But the user experience, too?

> Many product leaders specify the user experience inside the requirements document, regardless of the lifecycle. I've never understood that because as the customers see the product, they ask for changes in the user experience and the requirements.
>
> Instead of trying to specify the user experience, product leaders can specify the *goals* of that experience. Then, as everyone processes the internal and external feedback, the product leader can check the goals and see if the older goals align with the product evolution. Creating an incremental approach to the user experience works exactly the same way as the architecture. See Section 7.2.1: Coherent Product Architectures Emerge from the Work on page 89. The user experience emerges from the work.

That early and often delivery has other benefits. The team doesn't just learn from internal or customer feedback. The team also can take a little time to learn from how they worked and how they felt about that work.

Because agile teams limit their WIP and collaborate, they can create short feedback loops for delivery and learning. Or, they use their short feedback loops to reduce WIP and increase collaboration. That's a restatement of Little's Law.

Agile teams also visualize their work with a team board and with value stream maps.

7.1.1. Team Boards Visualize Progress

Since so many teams start their agile journey with Scrum, they often start with a three-column Scrum board as in the next figure.

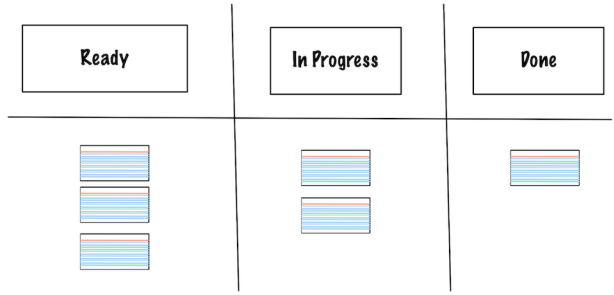

Figure 22. Three-Column Scrum Board

That board is terrific if you can keep your cycle time low and if the team collaborates on the work. Notice that this board has only two items in the In Progress column. So this team is collaborating and finishing.

But too often, I see teams who are supposed to use an agile approach but don't have the necessary people on their team, such as enough testers. Or, the team is also supposed to do production support.

Whatever the issue, their work gets stuck somewhere in the In Progress column. And they don't know where.

That's when teams might want a different board to explicitly show how work flows through their team. Six months ago, one of my clients used a Kanban board like the one in the next image to show their system of work.

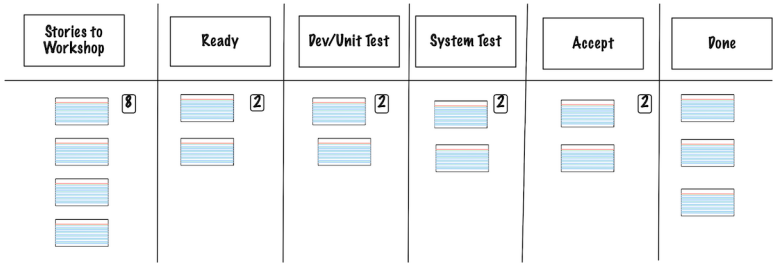

Figure 23. Multiple Column Kanban Board

One team on its agile journey decided to focus on lowering WIP in

their current state. They had a significant number of production support interruptions from a previous release. They wanted to integrate fixes with their current product development and avoid an Urgent or Expedite lane.

They chose to collaborate in twos or threes, either pairing or swarming on one item at a time. Only the product leader did not collaborate. That's because she was overloaded, supposedly working with two other teams as their product leader.

Read this board from the right side to the left to fully understand it.

This team decided to wait for four items to trigger a demo and a retrospective. Since their typical cycle time ran between one and three days, they tended to demo and retrospect every couple of weeks. (They also reviewed their throughput and tracked the age of all items as in Section 1.4.1: Measures Change in Flow Efficiency on page 7.)

To the left of Done is the Accept column. The reason the team has a WIP limit here is that the product leader will interrupt what she's doing to accept stories. She asked for that WIP limit so she doesn't get behind.

The remaining columns are also full. When I asked the team why they didn't have any slack in their system, one of them said, "We're still learning how to be most effective together. We keep experimenting with how we pair and swarm. We're thinking about mobbing/ensembling on the work, but not everyone is ready for that. But we know we need to keep our WIP low and we know we need to keep our cycle time low to manage our interruptions."

The only column that's not full is the Stories to Workshop column. Some teams create a cadence for workshopping stories. This team has a limit because their product leader wanted to workshop stories that they wouldn't add to a backlog for months. Instead of starting all that work, the Stories to Workshop column is a near-term roadmap or backlog.

This team controls how they work. Even so, the team has a full board, due to the previous culture of resource efficiency.

Six months later, that team changed its board. Because the product leader only works with this team, they no longer need WIP limits on the Accept column. In addition, the team rarely pairs, but works in triads. That allowed them to combine Dev/Unit Test and System Test into one column called Dev and Test.

Every agile team deserves a board that fits their current work and that they can use as a basis for experimentation. If your project has technical risks, your board might need to have a column for prototypes or experimentation. That's why the project, product, and organizational risks can change the approach a team can use.

Consider these questions for your board:

- Do we have enough information from our current board? If not, what else do we need?
- Do we need board-based WIP limits? Or, do we need column-based WIP limits?
- Do we have a policy about WIP limits? That is, do we allow ourselves to exceed them?

Boards aren't the only visualization that can help teams. Value stream maps can help a team visualize and isolate delays and bottlenecks.

7.1.2. Value Stream Maps Help Teams Visualize Problems

A value stream map shows two kinds of time: work time and wait time. The cycle time is the addition of the work time and the wait time, as in this figure.

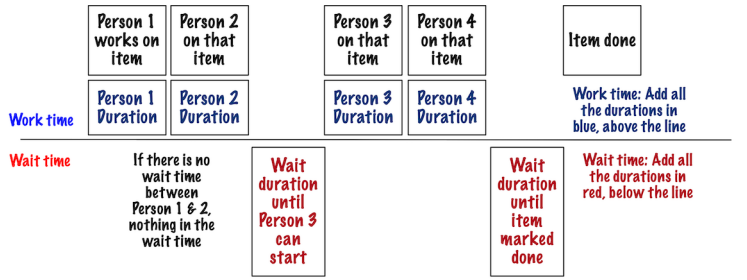

Figure 24. Blank Value Stream Map

When teams estimate, they often do a reasonable job estimating the work time. But very few people or teams estimate the wait time accurately. Teams who work cooperatively, but primarily alone, always have a longer cycle time than teams who collaborate.

When managers believe in resource efficiency, cooperative team members tend to be busy all the time. However, the team can't finish any *item* faster.

Collaborative teams work together, keeping their WIP low. That means they tend to have much less wait time for any of the items.

This is why teams need to see and manage their cycle times as a trend. When a team changes something—even if they only change their board—they might change their cycle time trends. When teams learn their tendencies for cycle time, they can choose how to work better. And they can offer 50%, 80%, or 90% confidence levels for future predictions.

Agile approaches are deceptively easy-looking—unless your organization has a culture of extreme resource efficiency. But fake agility thrives because of the necessary changes to the culture and the team characteristics.

Teams, managers—everyone has questions.

7.2. Questions About How to Make Agility Work

When I teach teams and managers about how to work in an agile way, everyone asks these questions:

- If we don't do a lot of upfront design, how can we create a coherent product architecture?
- What if it's too risky for our customers to take the next release?
- How can we possibly predict when we will be done?
- We need a project manager. Why does this notion of "agile team" not include a project manager?

Some teams also ask, "Should we use Scrum or a Kanban system and how do we decide?"

These are valid questions because non-agile teams managed these risks with iterative, incremental, and combination lifecycles for years.

I'll start with the idea of creating a coherent product architecture when the team iterates over the requirements and delivers incrementally.

7.2.1. Coherent Product Architectures Emerge from the Work

Many of the unplanned feedback loops arise from significant architecture, requirements, or user experience changes late in the project. Those changes cascade back and forth, where one feature during testing causes a requirements change that then causes an architecture change, that causes a user experience change, and so on.