Microsoft

Agile Project Management with Kanban

Best practices

Eric Brechner

Praise for *Agile Project Management with Kanban*

"I have been fortunate to work closely with Eric for many years. In that time he has been one of the most productive, consistent, and efficient engineering leaders at Xbox. His philosophy and approach to software engineering are truly successful."

—Kareem Choudhry, Partner Director of Software Engineering for Xbox

"Eric easily explains why Kanban has proven itself as a useful method for managing and tracking complicated work. Don't expect this book to be an overview, however. Eric channels his deep understanding and experiences using Kanban at Microsoft to help you identify and avoid many of the common difficulties and risks when implementing Kanban."

—Richard Hundhausen, President, Accentient Inc.

"Learning how Xbox uses Kanban on large-scale development of their platform lends real credibility to the validity of the method. Eric Brechner is a hands-on software development management practitioner who tells it like it is—solid, practical, pragmatic advice from someone who does it for a living."

—David J. Anderson, Chairman, Lean Kanban Inc.

"As a software development coach, I continuously search for the perfect reference to pragmatically apply Kanban for continuous software delivery. Finally, my search is over."

—James Waletzky, Partner, Crosslake Technologies

"Kanban has been incredibly effective at helping our team in Xbox manage shifting priorities and requirements in a very demanding environment. The concepts covered in Agile Project Management with Kanban *give us the framework to process our work on a daily basis to give our customers the high-quality results they deserve."*

—Doug Thompson, Principal Program Manager, Xbox Engineering

"An exceptional book for those who want to deliver software with high quality, predictability, and flexibility. Eric's in-depth experience in the software industry has resulted in a realistic book that teaches Kanban in a simple and easy way. It is a must-read for every software professional!"

—Vijay Garg, Senior Program Manager, Xbox Engineering

leadership often considers putting more resources on the project. How many people do you need to complete the project on time?

Before I show some calculations for right-sizing your team, there are a couple of important considerations:

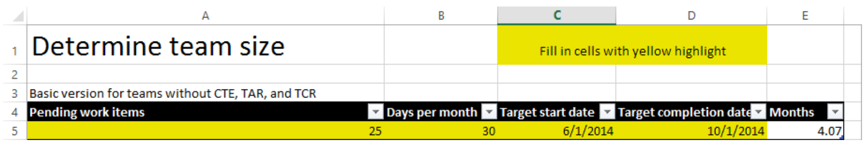
- The following calculations assume that team capacity grows linearly with team size. However, as Frederick P. Brooks Jr. pointed out in *The Mythical Man-Month* (1995), cross-team communication grows geometrically with team size. Thus, you can't double team size and expect double the throughput. Kanban's focus on smooth flow, visualizing work, and minimal meetings significantly reduces the impact of cross-team communication, but you still need to account for big team changes. Should the following calculations dictate that your team double or triple in size, consider refactoring your project into several teams that work together through a shared architecture and established interfaces.
- Adding new people to a team always slows it down before it speeds it up. New team members take a while to learn what they need to be productive, and veteran team members spend time answering questions and helping new team members acclimate. As a result, the time to add people is at the beginning or in the midst of the project—never at the end. (At the end of a project, you're better off slipping the release date or reducing the functionality.)

You can calculate the required size of your feature team in two ways:

- Use a basic approach before your team has measured its current task estimate (CTE), task add rate (TAR), and task completion rate (TCR). The basic approach requires only the same basic data that's used to determine your initial WIP limits in the Kanban quick-start guide (Chapter 2).
- Use an advanced approach after your team has detailed data about the current task estimate (CTE), task add rate (TAR), and task completion rate (TCR), as described earlier in the section "Track expected completion date."

Basic approach

Let's start with the basic approach. Say, you're putting together a team that has four months to release 25 new feature improvements to your product. (All the formulas and tables that follow are available in an Excel spreadsheet that you can download.)



Determine team size

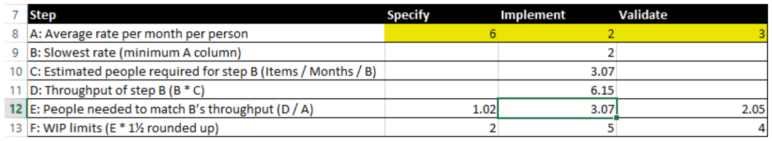
Fill in cells with yellow highlight

Basic version for teams without CTE, TAR, and TCR

Pending work items	Days per month	Target start date	Target completion date	Months
25	30	6/1/2014	10/1/2014	4.07

Note that the months are normalized to 30 days each, thus the count of 4.07 months. If you run a continuously deployed service or an operational team and just want to size your team to fit the flow of incoming requests, enter the number of requests per month as the count of pending work items, and use the start and end dates of a typical month.

Next, let's say you've got the same kind of team I used as an example in the Kanban quick-start guide (Chapter 2). Here's a worksheet showing the team-size calculation.



	Step	Specify	Implement	Validate
8	A: Average rate per month per person	6	2	3
9	B: Slowest rate (minimum A column)		2	
10	C: Estimated people required for step B (Items / Months / B)		3.07	
11	D: Throughput of step B (B * C)		6.15	
12	E: People needed to match B's throughput (D / A)	1.02	3.07	2.05
13	F: WIP limits (E * 1½ rounded up)	2	5	4

For the example:

- **A** On average, each analyst can specify roughly six items per month, each developer can implement roughly two items a month, and each tester can validate roughly three items per month. (You can use a shorter time frame by changing the number of days per month.)
- **B** Implementing is the slowest step (two items per month per person).
- **C** We want to estimate the number of developers required to implement 25 work items in 4.07 months. Since the developers can implement two items per month, you need 25 / 4.07 / 2 = 3.07 developers.
- **D** The throughput is 2 * 3.07 = 6.15 items per month (the extra hundredth is the result of Excel storing all these data values with higher precision).
- **E** Dividing that throughput by the average rates for each step gives you the people needed for each step (1.02 analysts, 3.07 developers, and 2.05 testers).
- **F** You calculate the WIP limits as in the Kanban quick-start guide (Chapter 2) by adding a 50 percent buffer to the people totals and rounding up (2 for Specify, 5 for Implement, and 4 for Validate).

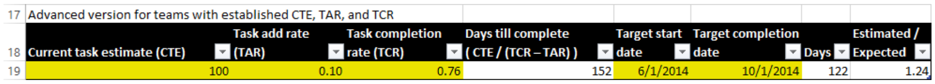
As in the Kanban quick-start guide, the WIP limits from the worksheet are starting values. The limits should be adjusted to maximize team output and agility. Check the Kanban quick-start guide's "Troubleshooting" section for when and how to adjust WIP limits for the best outcomes.

The people counts are also approximate starting values, which is why I left them as decimals. Once you form your team and start work, you can track your expected completion date over time and fine-tune the team as needed.

Advanced approach

The basic approach uses average work item rates per person per step to calculate the necessary team size. As I discussed in the estimation portions of this chapter, work item size has a great deal of variability. You get better estimates by breaking down work items into tasks and measuring actual task completion rates.

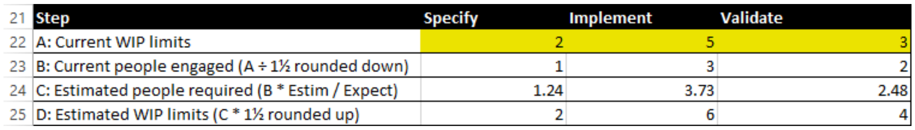
If you have the current task estimate (CTE), task add rate (TAR), and task completion rate (TCR), as described in the "Track expected completion date" section, you can estimate the proper team size with a bit more confidence.



17	Advanced version for teams with established CTE, TAR, and TCR							
18	Current task estimate (CTE)	Task add rate (TAR)	Task completion rate (TCR)	Days till complete (CTE / (TCR – TAR))	Target start date	Target completion date	Days	Estimated / Expected
19	100	0.10	0.76	152	6/1/2014	10/1/2014	122	1.24

For a comparison with the basic approach, I set the current task estimate to 100 (four tasks per basic-example work item), used the task add rate and task completion rate from the previous section, and the same 4.07 month period as for the basic example (122 days). I use days instead of months because task add rate and task completion rate are measured in days.

The estimated number of days needed to complete 100 tasks is 152, but the number of days available is 122. The ratio between the estimated and expected number of days is 152 / 122 = 1.24—that's the multiple of team members we need.



21	Step	Specify	Implement	Validate
22	A: Current WIP limits	2	5	3
23	B: Current people engaged (A ÷ 1½ rounded down)	1	3	2
24	C: Estimated people required (B * Estim / Expect)	1.24	3.73	2.48
25	D: Estimated WIP limits (C * 1½ rounded up)	2	6	4

In this example:

- **A** Rather than relying on the number of current team members assigned, who may not be fully dedicated to the team with the 0.76 task completion rate, we use the WIP limits that control flow.
- **B** We divide the limits by 1.5 and round down to determine the number of people completing 0.76 tasks a day. (That's the inverse of our algorithm to get the WIP limits.)
- **C** We then multiply by 1.24, the ratio between the estimated and expected number of days. This results in estimates for the number of people needed to complete the 100 tasks in 122 days.
- **D** Adding 50 percent to those people totals and rounding up gives you each step's WIP limit (2 for Specify, 6 for Implement, and 4 for Validate).

Note that the WIP limits and people estimates differ between the advanced approach and the basic approach. Two reasons account for this:

- If you assume four tasks per work item, the throughput for the basic-approach example would be 6.15 * 4 = 24.6 tasks per month, for a task completion rate of 24.6 / 30 days = 0.82 tasks per day. That's slightly higher than 0.76 tasks per day we used in the advanced approach example.
- The basic approach doesn't account for the task add rate (0.1 tasks per day). Over 122 days, that's 12.2 additional tasks to complete.

As with the basic approach, WIP limits and people counts derived through the advanced approach are just starting values. You should adjust them as your team does the real work and gets real feedback data.

Because Kanban continuously provides you with updated data on your project, it's easy to adjust your estimates, WIP limits, and team sizes to hit your deadlines. In time, your delivery of customer value becomes so predictable that estimation is easy and scheduling accuracy is less of a concern.

Checklist

Here's a checklist of actions to ensure that you hit your deadlines:

- ❑ Collect product-improvement ideas from your customers and your business, putting descriptive names for each idea on its own card.
  - Include an indicator on each card for the source of the idea (helps with ordering and recognition of the cards).
- ❑ Place the cards into one of four buckets: must-have and wouldn't release without (pri 0: the minimum viable product), should have (pri 1), like to have (pri 2) and nice ideas (pri 3).
- ❑ Order a subset of the cards in the Backlog column of your signboard, selecting first from the minimum viable product and then augmenting with related or additional lower-priority cards as needed.
- ❑ As needed, let your leadership, customers, and internal and external partners know when to expect results.
  - Estimate the number of tasks needed to complete work items, preferably using a Wideband Delphi method such as planning poker.
  - Calculate your task completion rate (divide the number of tasks that are done with their final steps within any two-week to four-week period by the number of days in that period).
  - Report the estimated date by dividing the total estimated number of tasks in consideration by the task completion rate, and adding that number of days to the current date.
- ❑ As needed, track and report your expected completion date for your leadership, customers, or partners.
  - Compute your task completion rate (TCR), current task estimate (CTE), and task add rate (TAR).
  - Calculate your expected completion date by adding the result of CTE / (TCR – TAR) to the current date.
- ❑ As needed, right-size your team to complete your project on time, using one of two methods.
  - Use a basic approach to calculate the number of people needed for each step, based on average rates per month per person for each step.
  - Use an advanced approach to calculate the number of people needed for each step, based on task completion rate (TCR), current task estimate (CTE), and task add rate (TAR).

This page intentionally left blank

CHAPTER 4

Adapting from Waterfall

This chapter is for people currently using a traditional Waterfall method for product development. If your team uses Scrum, please feel free to skip to the next chapter, "Evolving from Scrum."

Kanban is simple in structure and uses common terminology. As a result, a wide range of people can begin using it without significant trouble or prolonged explanation. As with any new method, it takes a few weeks to adjust to Kanban and a few months to master it. Nonetheless, even if you learned product development decades ago, you can quickly get started and feel productive with Kanban. After a couple of months, the increases in productivity, quality, predictability, and agility should be evident and measurable to your leadership and your team.

When I talk about "traditional Waterfall," I mean the practice of writing specs, implementing features, and performing validation in bulk (many features at once) over the course of milestones that often span months. I've experienced many variations of Waterfall at Microsoft, Boeing, and other places where I've worked.

This chapter will help you adapt your variation of traditional Waterfall to Kanban without much fuss or hassle. I've even included a rude Q & A listing questions that a blunt team member might ask, followed by pragmatic answers meant to reassure, build trust in the new approach, and clearly explain how to achieve great results with Kanban.

The topics covered are:

Introducing Kanban to a Waterfall team

A traditional Waterfall team is one that likely has members who've been doing product development for decades. Their habits were formed long ago and have served them well over the years. Although the products they build might be buggy initially, the members of the traditional Waterfall team know