Putting the 'flow' back in workflow with WIP limits

Work-in-progress limits aren't meant to actually limit your progress. Quite the opposite, in fact.

BY DAN RADIGAN

What are WIP limits?

In agile development, work in progress (WIP) limits set the maximum amount of work that can exist in each status of a workflow. Limiting the amount of work in progress makes it easier to identify inefficiency in a team's workflow. Bottlenecks in a team's delivery pipeline are clearly visible before a situation becomes dire.

Why are WIP limits important?

So now you're thinking, "Tell me more!" (Well, I hope you are.)

WIP limits improve throughput and reduce the amount of work "nearly done", by forcing the team to focus on a smaller set of tasks. At a fundamental level, WIP limits encourage a culture of "done." More important, WIP limits make blockers and bottlenecks visible. Teams can swarm around blocking issues to get them understood, implemented, and resolved when there is a clear indicator of what existing work is causing a bottleneck. Once blockages are removed, work across the team begins to flow again. These benefits guarantee that increments of value are delivered to customers sooner, making WIP limits a valuable tool in agile development.

> Also, multitasking is deceptively time-intensive.

RELATED TUTORIAL

Learn kanban with Jira Software

Try this tutorial →

SUBSCRIBE

Sign up for more articles

Email

email@example.com

Subscribe

During development, it's easy to think "Oh, I'll pause on this one issue while I begin work on another." Having two issues open means context switching between two different things or transferring work between teammates. Ramping down on one issue and up on another isn't free–it takes time and degrades focus. It's almost always better to work through the original issue rather than starting–and not completing–new work. In other words, WIP limits discourage us from impeding our own flow.

Finally, WIP limits point out areas of chronic idleness or overload. They help the team see inefficiencies in the entire process rather than just the particular area in which they work.

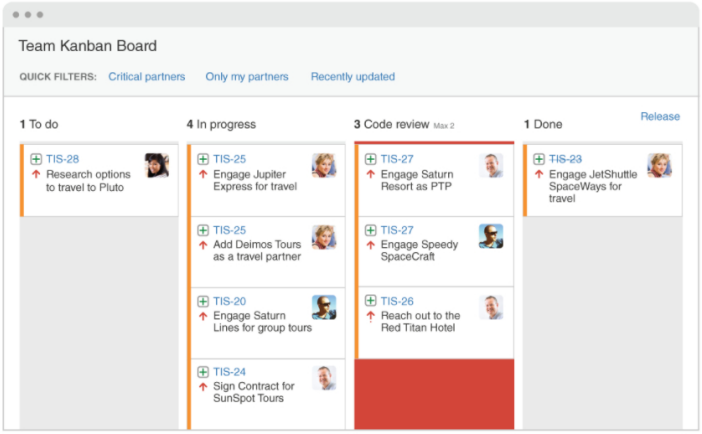
PRO TIP:

For teams new to WIP limits, they will feel awkward. Take the time to discuss them in the first few iterations. Understand when and why the team hit the WIP limits. Resist the temptation to arbitrarily adjust them at first. If a breach becomes consistent, that's a sign that either the WIP limit is too restrictive or the team's process is inefficient.

Using WIP limits on agile teams

Now that you're sold on their value, let's get down to brass tacks.

When rolling out a new workflow, make a team decision to determine the WIP limits for each status. We recommend setting WIP limits after monitoring the average number of work items in each status for a few sprints. Below is a sample agile board with WIP limits used by a typical software development team.



Above, a WIP limit has been set on code review. Since the column is exceeding its limit, the background has turned red.

Since there's nothing left to do once an issue reaches done, there is no need for a WIP limit there.In the board above, "Ready for dev" signifies that the story has been fully vetted by the product owner and team. The

development team pulls work from "ready for dev" into "in progress" as they start on work items. As a best practice, it's important to keep enough work in the "ready for dev" status so that each member of the development team remains fully utilized. By keeping only just enough stories in the "ready for dev" state, the product owner doesn't get too far ahead of the game when it comes to fleshing out requirements, and the program becomes more responsive to change.

The "in progress" status lists work that's under active development. The goal of WIP limits in this case is to ensure that everyone has work to do, but no one is multitasking. In the board above, the limit for "in progress" items is 4, and there are currently 3 items in that state. This tells the team they've got capacity to take on more work. As a best practice, some teams set the maximum WIP limit below the number of team members. The idea is to bake in room for good agile practices. If a developer finishes an item, but the team is already at their WIP limit, they know it's time to knock out a few code reviews or join another developer for some pair programming.

The "code review" status indicates stories that have been fully written but need to be reviewed before merging into the code base. Timely code reviews are a best practice that establish quality, get new innovation out to market faster, make merges easier by reducing open branches, and spread knowledge across the engineering team. Items in this state should be acted on urgently, for a few reasons:

- The code doesn't rot as team members check in new code
- The developer doesn't lose the context he or she gained in writing the original code
- The feature can be merged into the main branch for release

WIP limits guarantee un-reviewed code doesn't pile up.

Note that in the agile board above, the team has too many code reviews, so the column has turned red to indicate that.

> ANTI-PATTERNS TO WATCH FOR:
>
> - WIP limits are raised as needed so the team doesn't hit them anymore. ("Debt ceiling," anyone?)
> - Everyone has a large "background task" on their plate to mask times when they'd otherwise be idle.
> - Team members sit idle waiting for more work to pull in, rather than swarming on bottlenecks.
> - Throwing more person hours at persistent bottlenecks is preferred over improvements in

> engineering practices or team processes.

4 goals for agile teams using WIP limits

Like any new activity, WIP limits may feel awkward at first. The goal here is to optimize the team over the medium-term, and the short-term awkwardness is actually a good thing. It's causing the team to feel some pain points in their process. After the team uses their WIP limits for a few weeks, make adjustments as needed. Resist the temptation to raise a WIP limit just because the team keeps hitting it. Seize that opportunity to increase capacity–ideally, by educating the team and giving each member new skill sets or making some aspect of the development process more efficient.

Goal 1: Size individual tasks consistently. When breaking down requirements and user stories, it's important to keep individual tasks to no more than 16 hours of work. Doing so increases the team's ability to estimate confidently, and it helps prevent bottlenecks. Nothing slows down a team and aggravates WIP limits like a big work item clogging up the pipeline.

> PRO TIP:
>
> When work in progress limits are working for the team, an issue's cycle time will drop. Cycle time is the amount of time it takes to complete an issue. Check out our page on agile metrics to learn more.

Goal 2: Map WIP limits to the team's skills. The above example assumes team members have similar skill sets. If your team has specialists on it, work in progress limits may differ when the specialist is involved. Create a status specific for the specialist's work. If bottlenecks occur in that status, use the opportunity to educate other team members to add additional capacity for the specialist's skill sets and increase flow across the entire team.

Goal 3: Reduce idleness. When a team member has some downtime, encourage them to help an upstream or downstream team member. They'll contribute to the overall productivity of the team, and learn something along the way!

Goal 4: Protect a sustainable engineering culture. Work in progress limits do not mean developers need to rush through work to avoid work overload in a particular status. They are meant to support solid agile engineering practices that protect the quality of the product and health of the code base.

SHARE THIS ARTICLE

DAN RADIGAN

Agile has had a huge impact on me both professionally and personally as I've learned the best experiences are agile, both in code and in life. You'll often find me at the intersection of technology, photography, and motorcycling.

TUTORIAL

Learn kanban with Jira Software

Step-by-step instructions on how to drive a kanban project, prioritize your work, visualize your workflow, and minimize work-in-progress with Jira Software

Try this tutorial →

ARTICLE

Kanban vs Scrum

Discover if Kanban or Scrum is a better framework for your agile team. Learn the key differences between the two frameworks.

Read this article →

Agile Topics

Agile project management
Scrum
Kanban
Design
Software development
Product management
Teams
Agile at scale
DevOps

Sign up for more agile articles and tutorials.

Email

email@example.com

Subscribe

Up Next
Kanban vs Scrum →