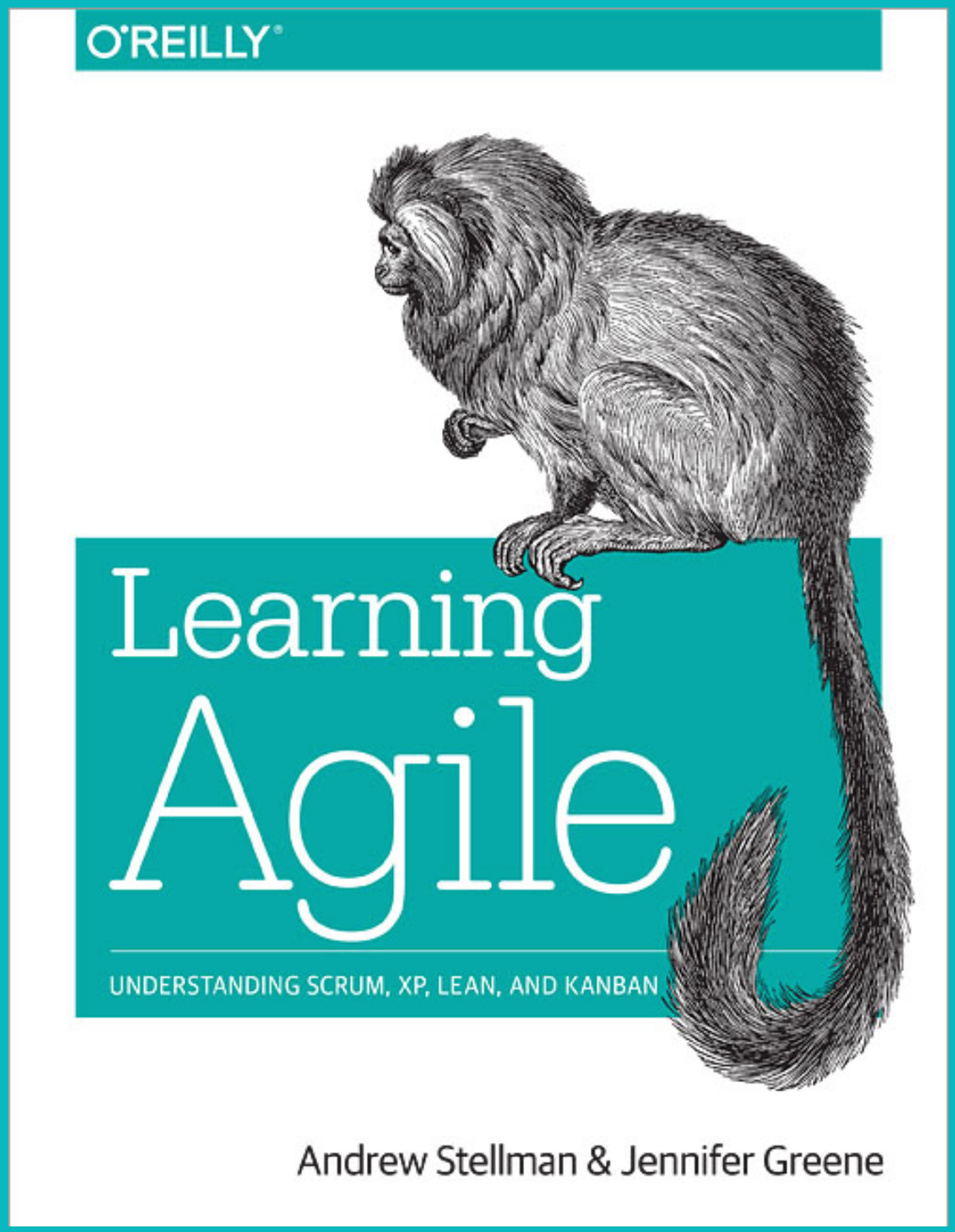
Download this deck: http://bit.ly/learning-agile-antipatterns

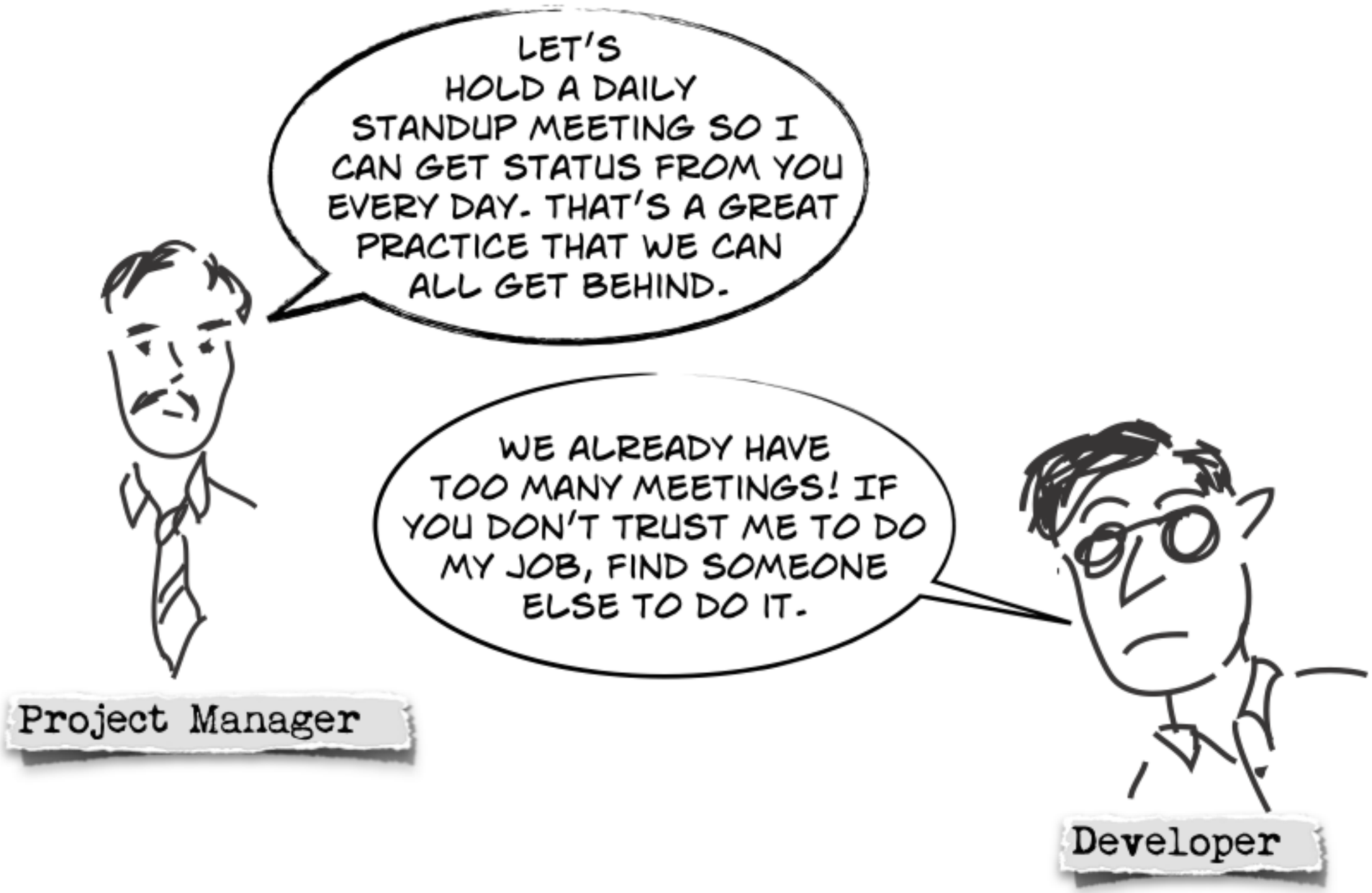


Agile Antipatterns

What does "better-than-not-doing-it" mean for agile teams?

http://www.stellman-greene.com

Practices are more or less effective depending on the team's mindset

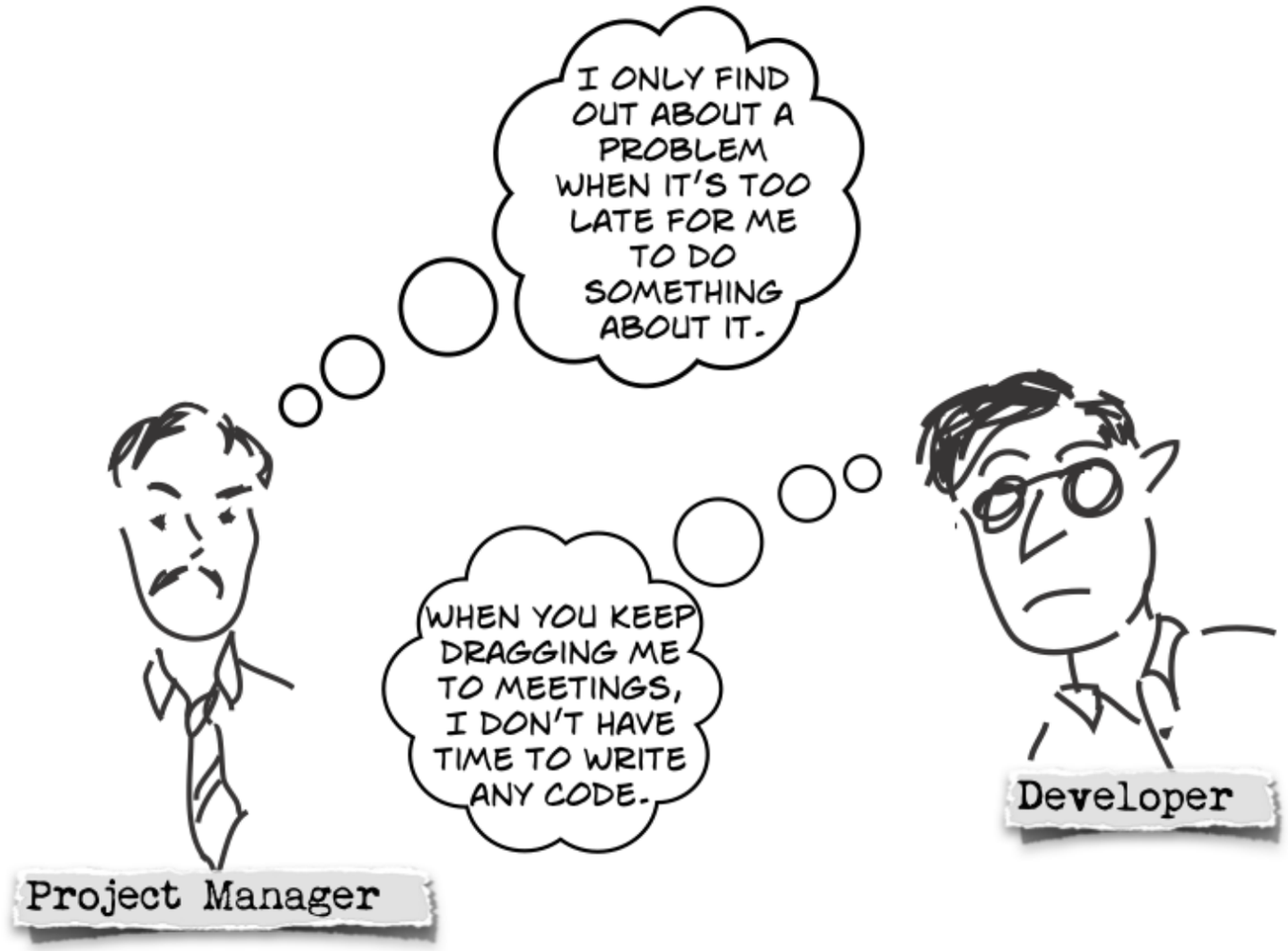


They miss the point... but it's still worth holding the meeting

The project manager uses the daily standup to get status updates on his plan and give team members their next assignments.

The developer wants to get back to coding, so he gives his update and spends the rest of the meeting looking at his phone.

The meeting is not nearly as effective as it could be, but it's still worth doing. The team got better-than-not-doing-it results.



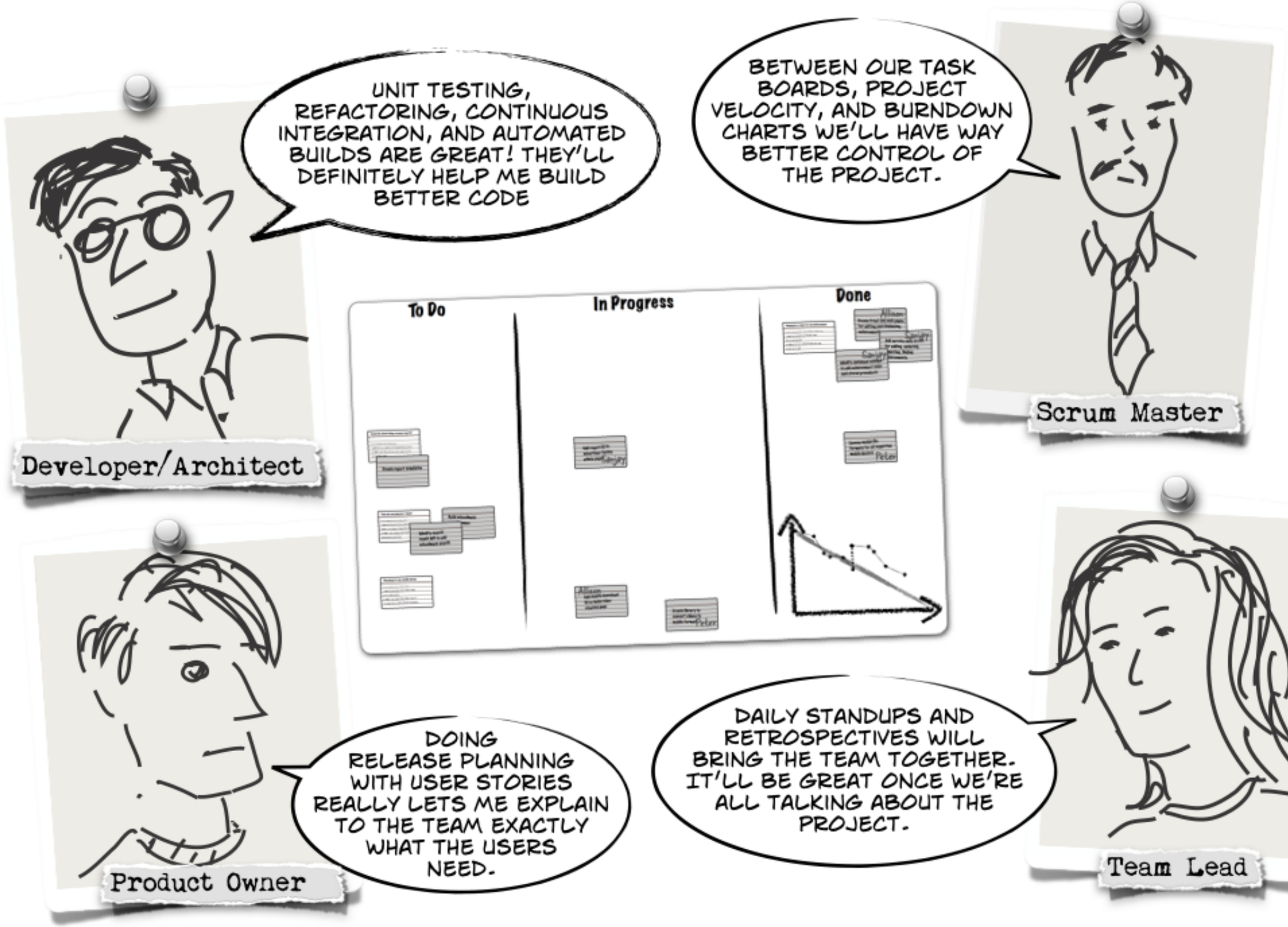
When team members understand the principles behind the daily standup practice, they get more out of doing it

What if the developer and project manager had a different mindset?

The project managager doesn't think of it as his plan, he thinks of it as the plan everyone on the team worked together to create.

The developer does more than just give status. He has (and shares) opinions on the whole project, and the daily standup becomes important to him.

Most team members focus on practices that directly help them do their jobs



When the team gets the principles, they get more out of the practices

This is about getting everyone on the team talking to their teammates and understanding their perspectives instead of just hyper-focusing on one aspect of the project.

By keeping everyone focused on the end goals instead of stumbling on intermediate problems, the team an keep the project moving forward.

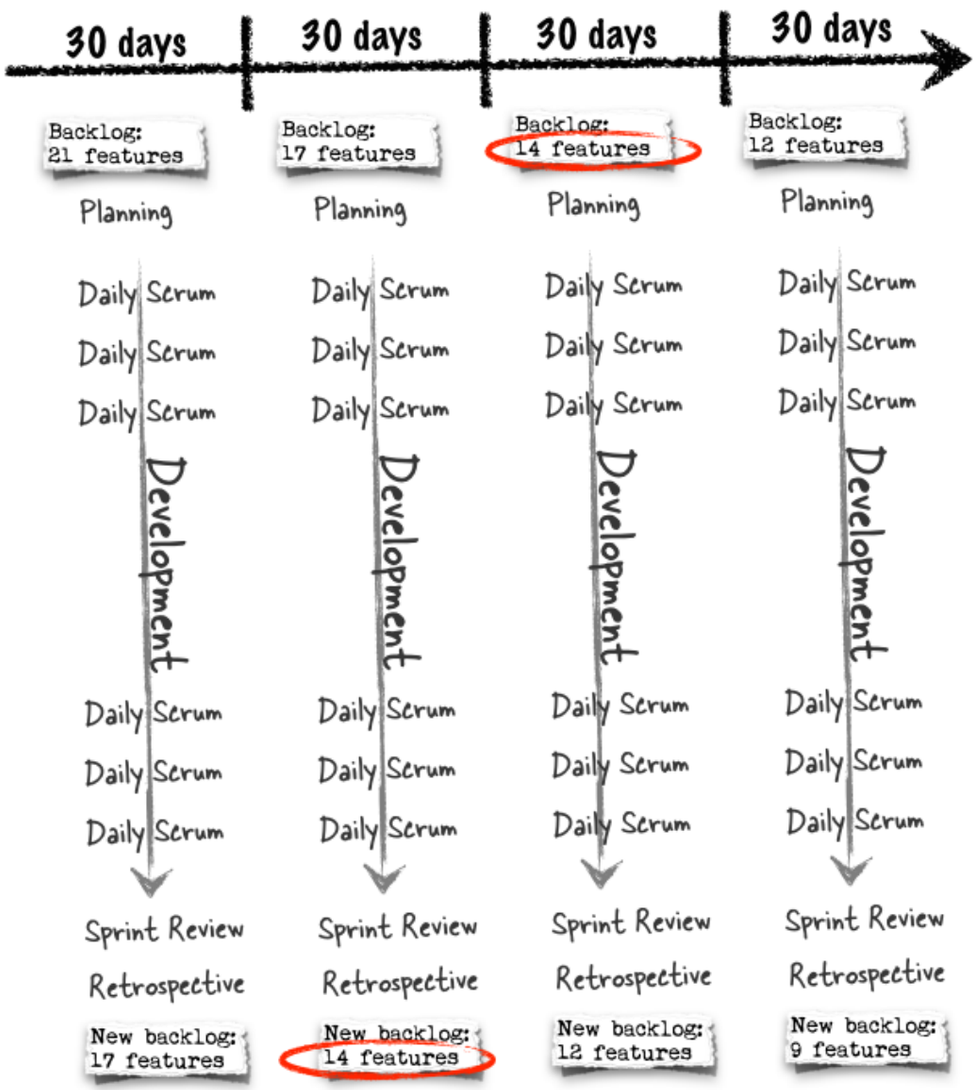
The Agile Manifesto

* Individuals and interactions over processes and tools
* Working software over comprehensive documentation
* Customer collaboration over contract negotiation
* Responding to change over following a plan

It's easy for a hyper-focused team to lose track of what the users actually need. This makes sure that the users' perspectives and ideas are genuinely represented.

...because working the wrong plan causes the team to build the wrong software.

What is Scrum?



- There are three main roles on a Scrum project: Product Owner, Scrum Master, and team member.

- The team (including the Product Owner) maintains a backlog of features and requirements that need to be built, organized by value and difficulty.

- The software is built using timeboxed iterations called sprints. At the start of each sprint, the team does planning to determine which features from the backlog to build.

- Every day, the team holds a short face-to-face meeting called a Daily Scrum to update each other on the progress they've made, and the roadblocks ahead.

- The Scrum Master keeps the project rolling by helping the team identify and remove roadblocks. At the end of the sprint, working software is demonstrated, and the team holds a retrospective to figure out lessons they've learned so they can improve.

Scrum is more than practices and roles: it has values, too

Practices

- **Project backlog**
- **Sprints, sprint planning, sprint backlog**
- **Daily Scrum meeting**
- **Task boards and burn-down charts**
- **Sprint and project retrospectives**
- **User stories, story points, velocity**

Roles

- **Product Owner**
- **Scrum Master**
- **Team**

Values

Commitment: Each person is committed to the project's goals

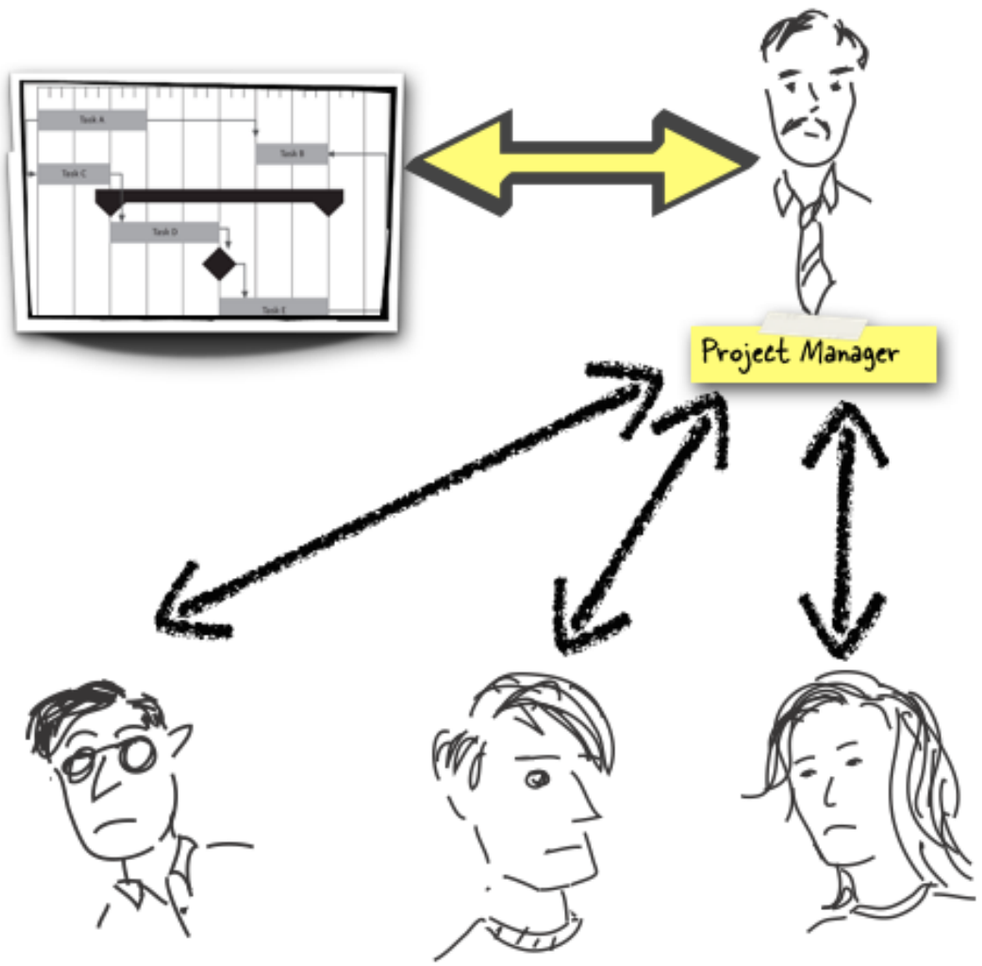
Respect: Team members respect each other

Focus: Everyone is focused on the work

Openness: Each team member is aware of the work everyone else is doing

Courage: Team members have the courage to stand up for the project

What does better-than-not-doing-it Scrum look like?



- Command-and-Control PMs maintain the schedule, get status updates from the team
- Team Leads assign work to the team members and they do what's required
- Requirements come in at the start of the sprint and any changes are pushed to the next sprint
- Iterations feel like mini-waterfall projects with many of the same pitfalls
- The whole Scrum adoption somehow feels "empty" – like everyone's just going through the required motions, but not much has really changed for the project.

What is XP?

Practices

- **Test-First Programming**
- **Pair Programming**
- **10-Minute Build**
- **Continuous Integration**
- **Incremental Design**
- **Weekly Cycle**
- **Stories**
- **Slack**
- **Quarterly Cycle**
- **Sit Together**
- **Whole Team**
- **Informative Workspace**
- **Energized Work**

Values

Communication: Each team member is aware of the work everyone else is doing.

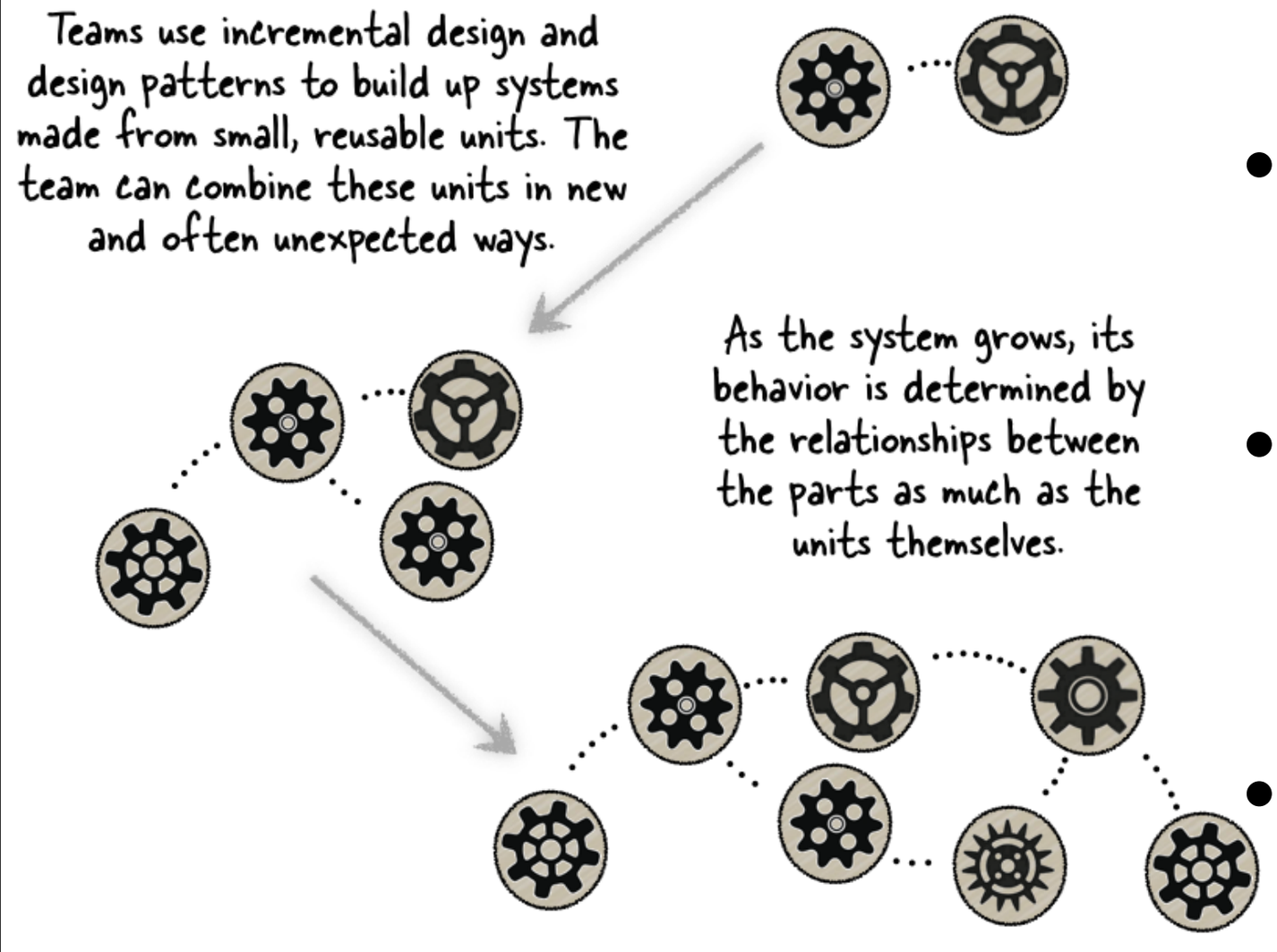
Simplicity: Developers focus on writing the most simple and direct solutions possible.

Feedback: Constant tests and feedback loops keep the quality of the product in check.

Respect: Every team member is important and valuable to the project.

Courage: Each team member is focused on making the best choices for the project, even if it means having to discard failing solutions or approach things differently.

Incremental design helps teams build products that are easy to change



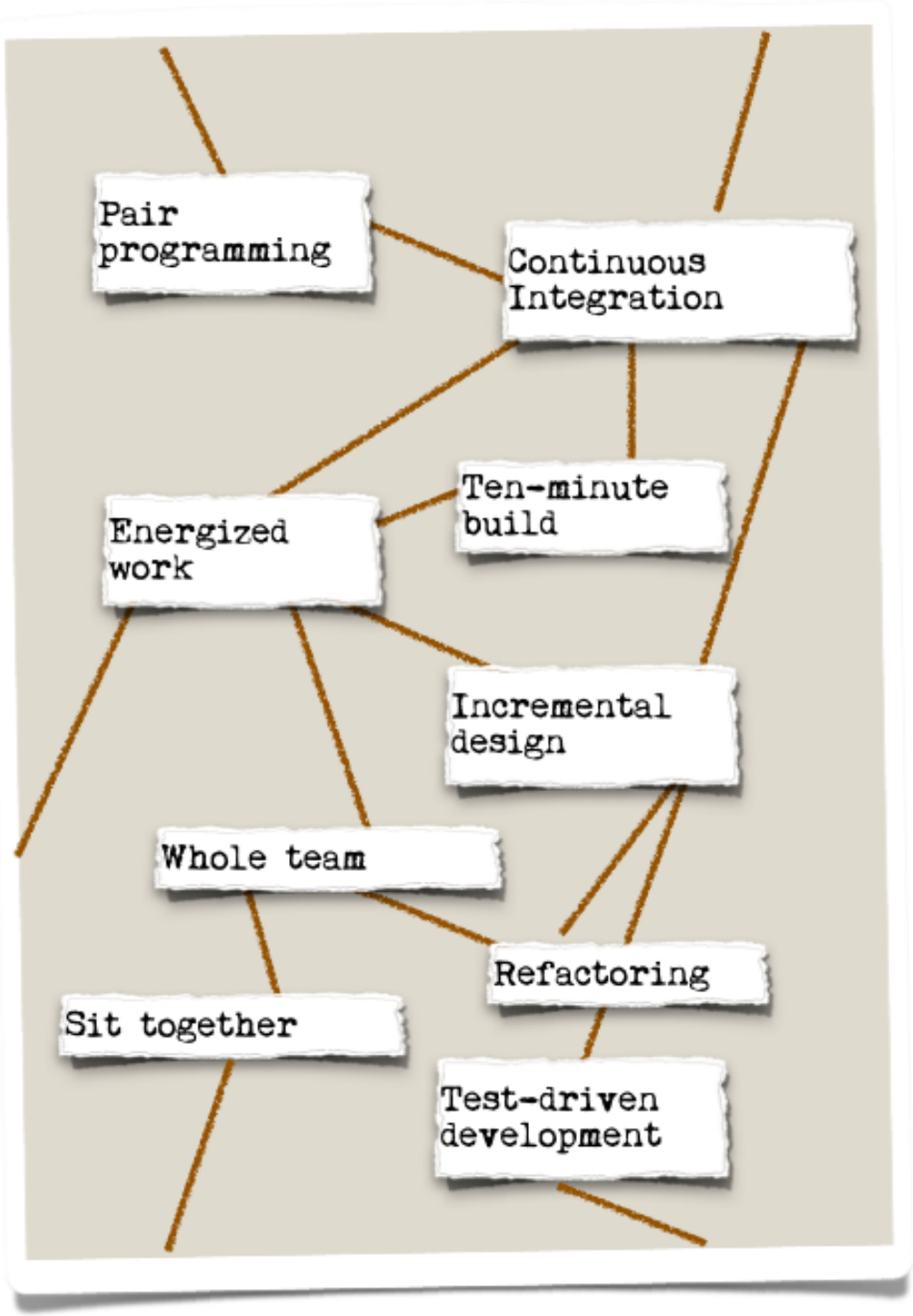
- **Even highly skilled teams can run into code problems when they have bad habits.**

- **XP teams spot code smells (or antipatterns in their code) to avoid complexity and keep their design simple.**

- **When the team has developed the habits that lead them to build decoupled code made up of small, solid, independent units, they don't need a complete design up front.**

- **But it only works when every developer truly believes in making design and code decisions at the last responsible moment.**

What does better-than-not-doing-it XP look like?

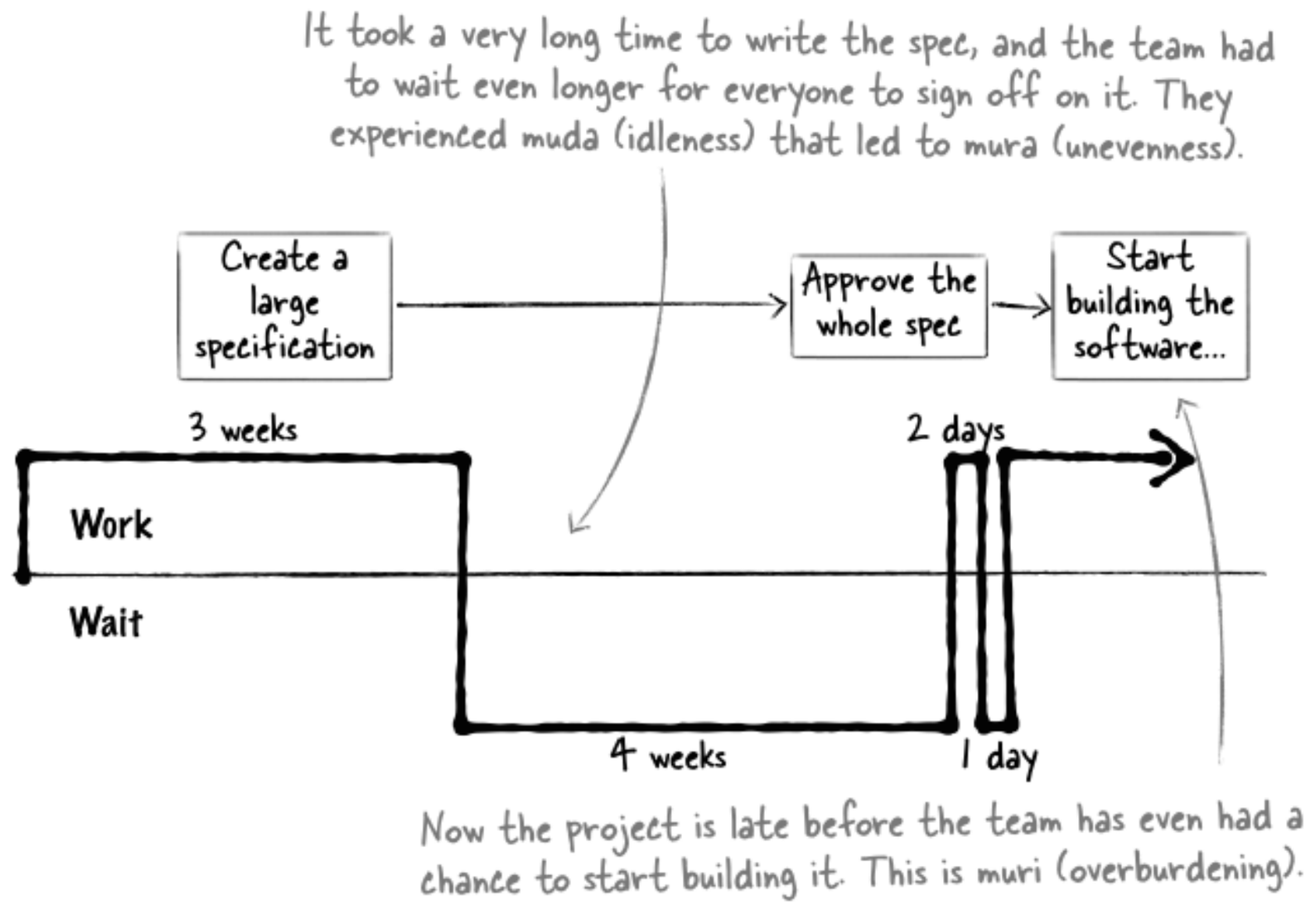


- Every practice seems to have a "sensible" alternative that the team likes better, which takes away from the ecosystem effect of the practices.
- People rarely pair up because it seems like a waste of time, the team rarely writes tests first because it's hard to do.
- The team puts in late nights and weekends often.
- Short term deadlines force the team to do highly-coupled, monolithic designs.
- Refactoring is seen as a nice-to-have and the team rarely gets to pay down its tech debt.

What is Lean?

Lean values

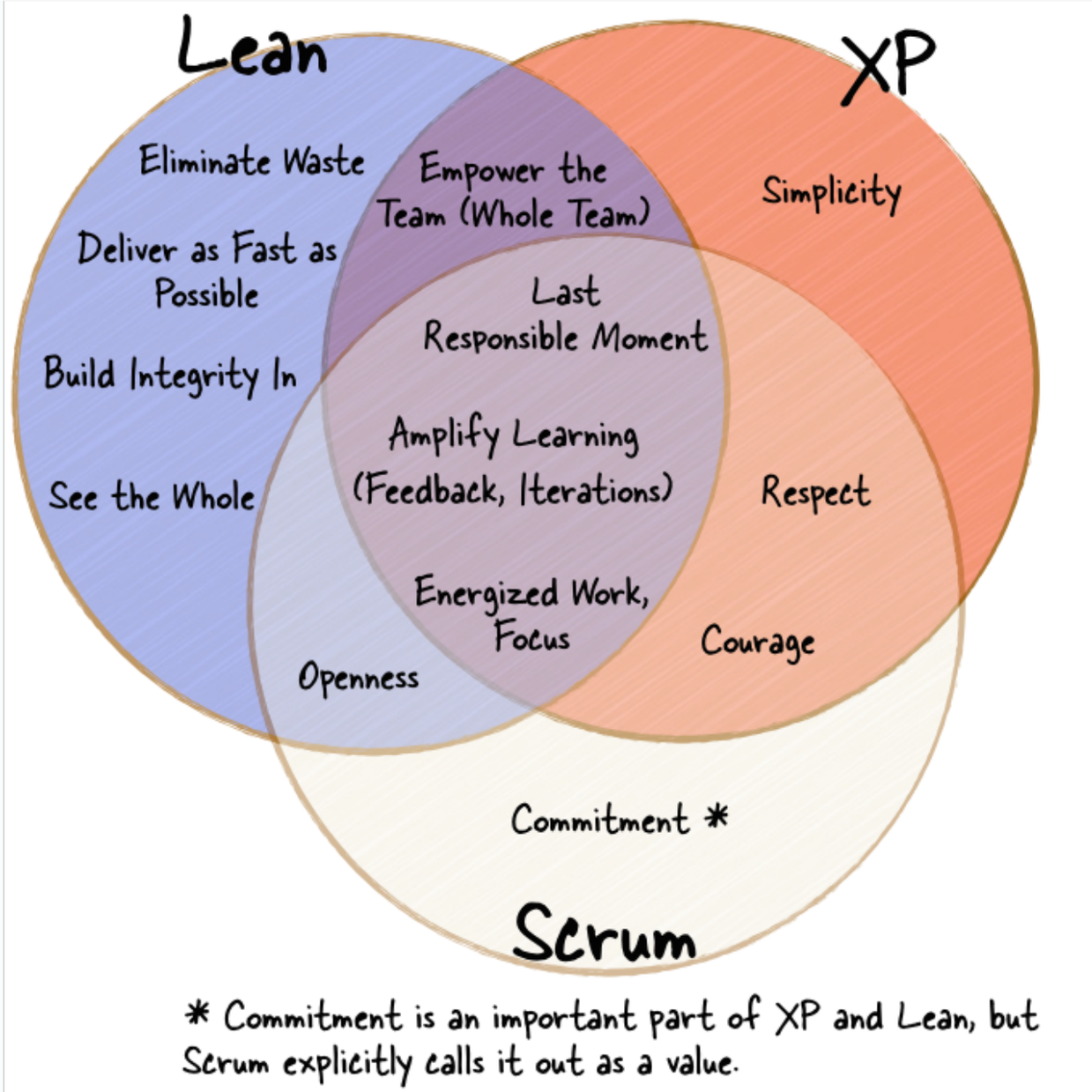
- **Eliminate waste**
- **Amplify learning**
- **Decide as late as possible**
- **Deliver as fast as possible**
- **Empower the team**
- **Build integrity in**
- **See the whole**



The three types of waste

- Muda (無駄), which means “futility; uselessness; idleness; superfluity; waste; wastage; wastefulness”
- Mura (斑), which means “unevenness; irregularity; lack of uniformity; nonuniformity; inequality”
- Muri (無理), which means “unreasonableness; impossible; beyond one’s power; too difficult; by force; perforce; forcibly; compulsorily; excessiveness; immoderation”

What is agile?

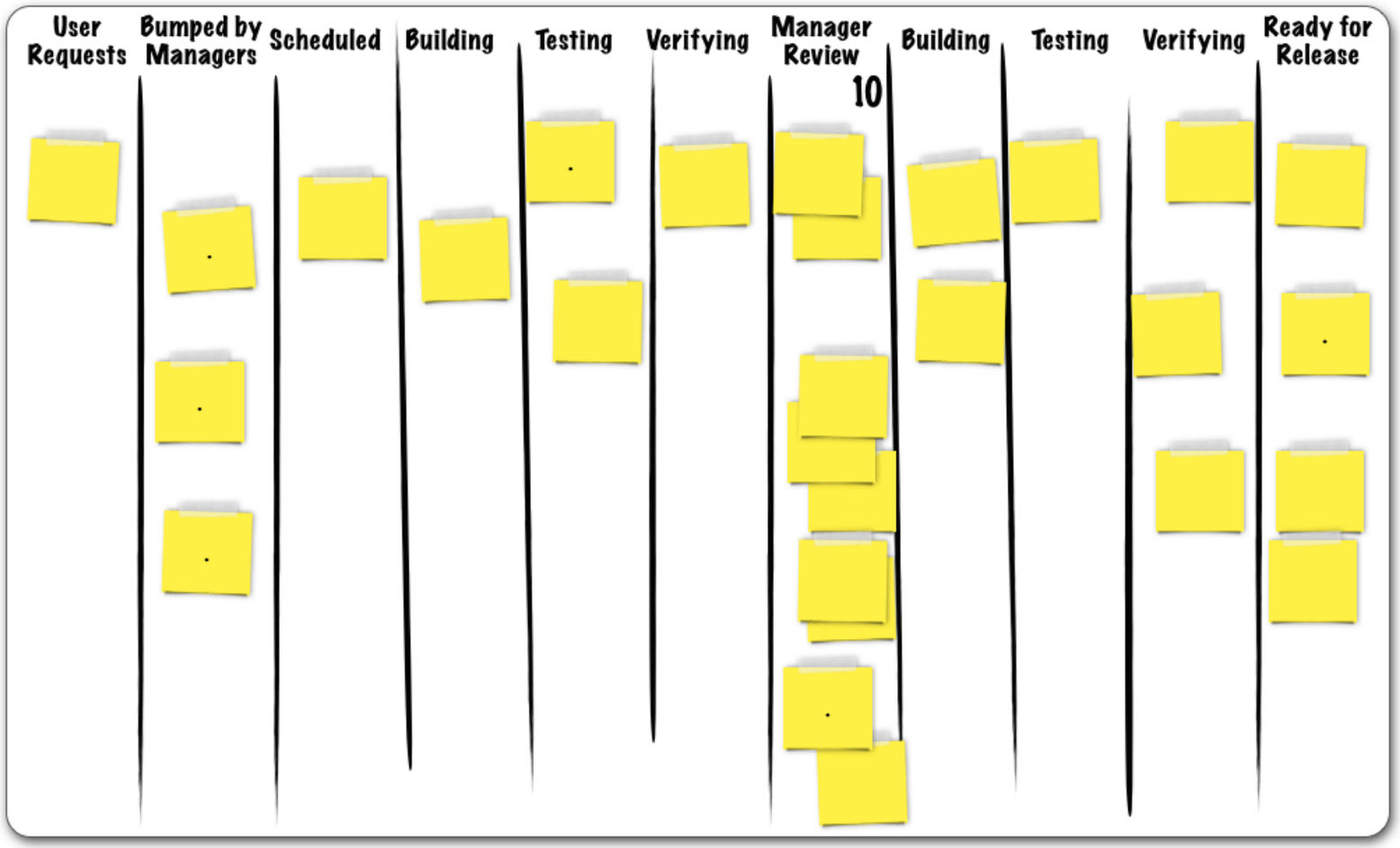


- Lean, XP, and Scrum are very different, but have a lot in common, and what they have in common helps us understand what agile is.
- Agile teams make decisions at the last responsible moment.
- Agile teams practices that create feedback loops (like iterations or a release cadence, or writing unit tests first).
- Agile teams create an energized working environment, and give each person on the team freedom to focus on one thing at a time.
- Agile teams give everyone a voice in how the project is run.
- The agile values and principles in the Agile Manifesto help us understand what it means to have an agile mindset.

What is Kanban?

Properties

- Visualize the workflow
- Limit work-in-progress (WIP)
- Measure and manage flow
- Make process and policies explicit
- Use models to recognize improvement opportunities

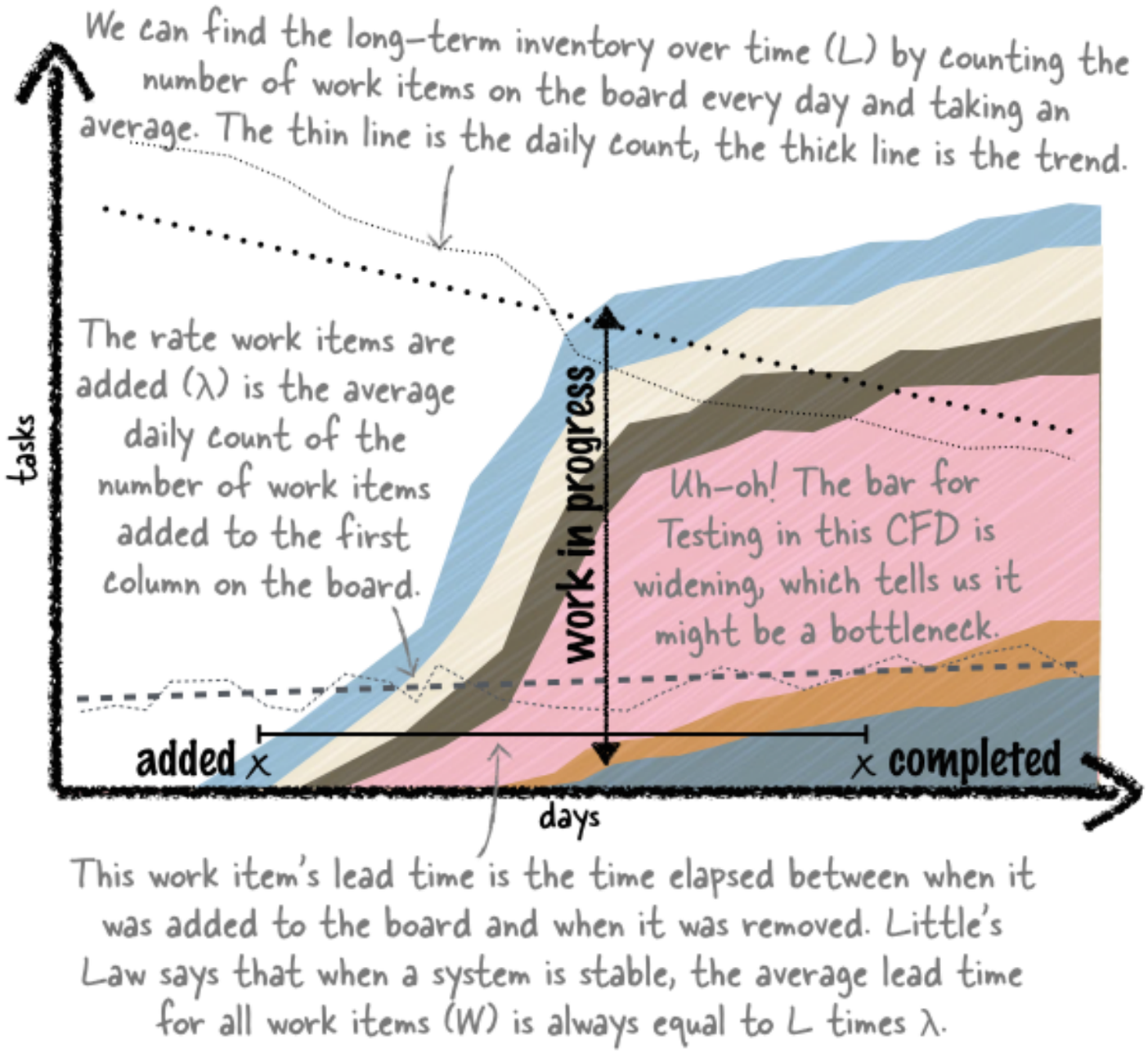


Kanban helps teams do process improvement by managing flow

- **Kanban teams visualize the workflow, and manage the flow of work through it.**

- **They use Kanban boards and cumulative flow diagrams to establish WIP limits.**

- **WIP limits ensure that work flows through the system as efficiently as possible.**



What does better-than-not-doing-it Kanban look like?

- The team doesn't understand that the values of the Lean mindset drive Kanban and make it work.

- They treat Kanban as a system for project management, not a way to continuously improve how the organization builds software.

- Kanban is implemented as an add-on to Scrum (eg. they assume that Kanban boards are task boards).

- But while Scrum protects the team by giving them power over the sprint backlog, Kanban externalizes it, but since management didn't agree to WIP limits or policies, they can overload the team and remove their focus.

Can you do better than better-than-not-doing-it results?

- ☐ Choose a methodology with values that match your team and company culture, and practices that address real problems.
- ☐ There are many paths to becoming agile, but there is no single, universal agile implementation that you can simply follow.
- ☐ Start with the practices, because a great way to learn a new way of thinking is to start acting differently.
- ☐ A practice that works really well for one team can utterly fail for another because they have a different attitude and mindset.