If productivity matters
(and it probably does)
then...

TEAM UP!

industrial logic

If productivity matters
(and it probably does ^not)
then…

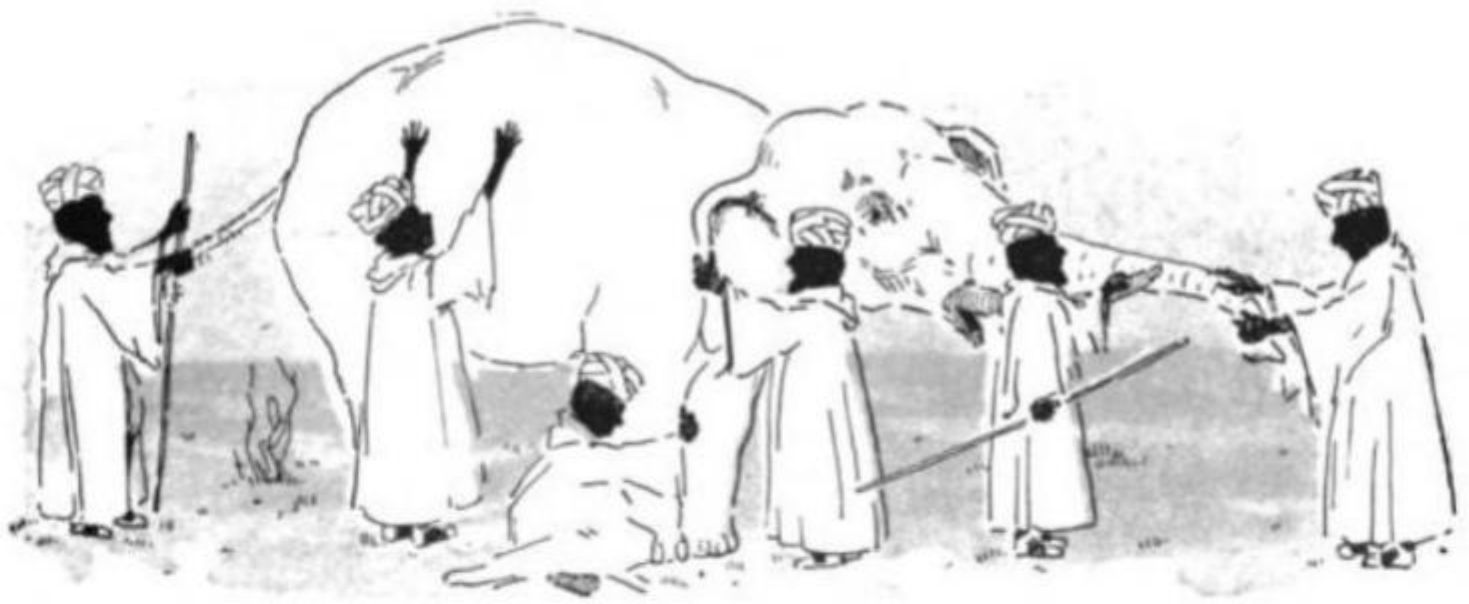
TEAM UP!

industrial logic

There are no villains in this piece

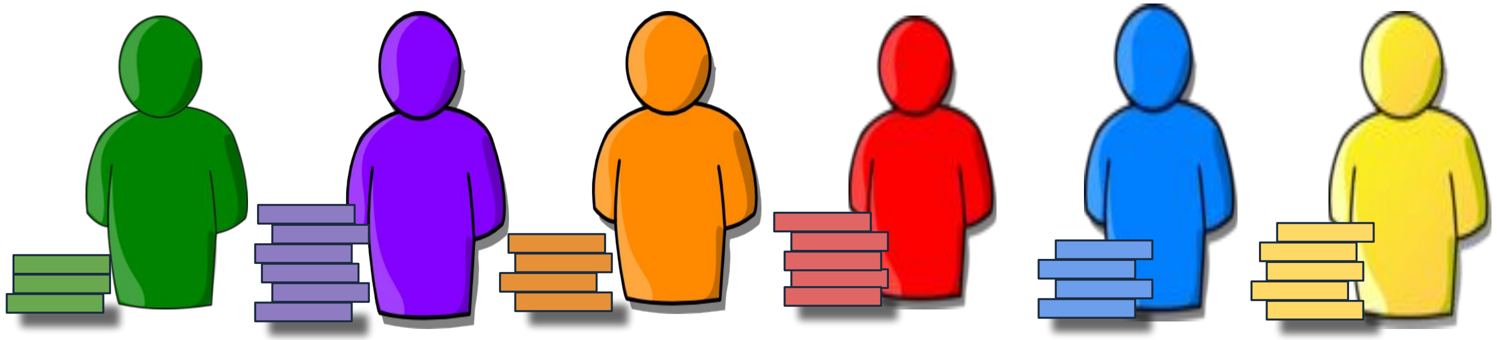
Almost all of the people
Almost all of the time
Are just trying to be helpful

Don Gray



Over-communicating As Coping

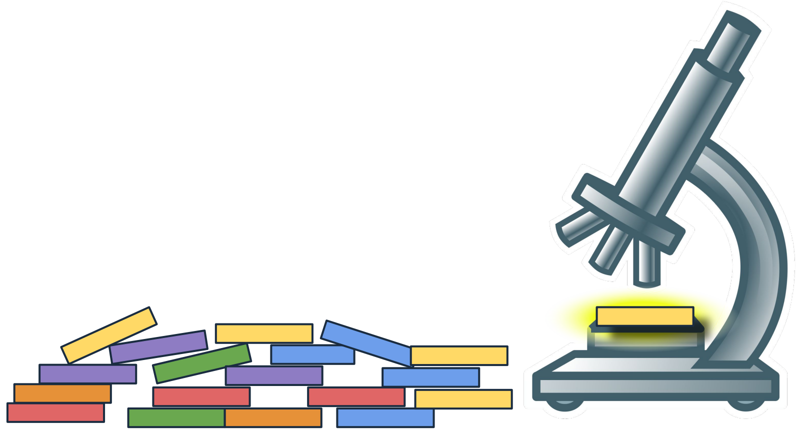
Individual Contributors



People have their own skill, knowledge, abilities, methods.

Work is split and assigned to the best individual for the job.

Inspections

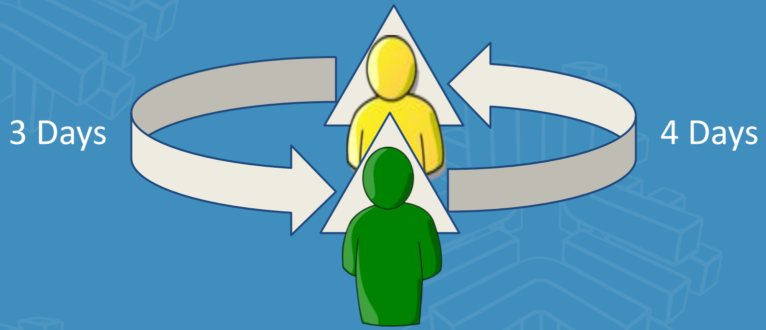


Each person has limited skills and knowledge.

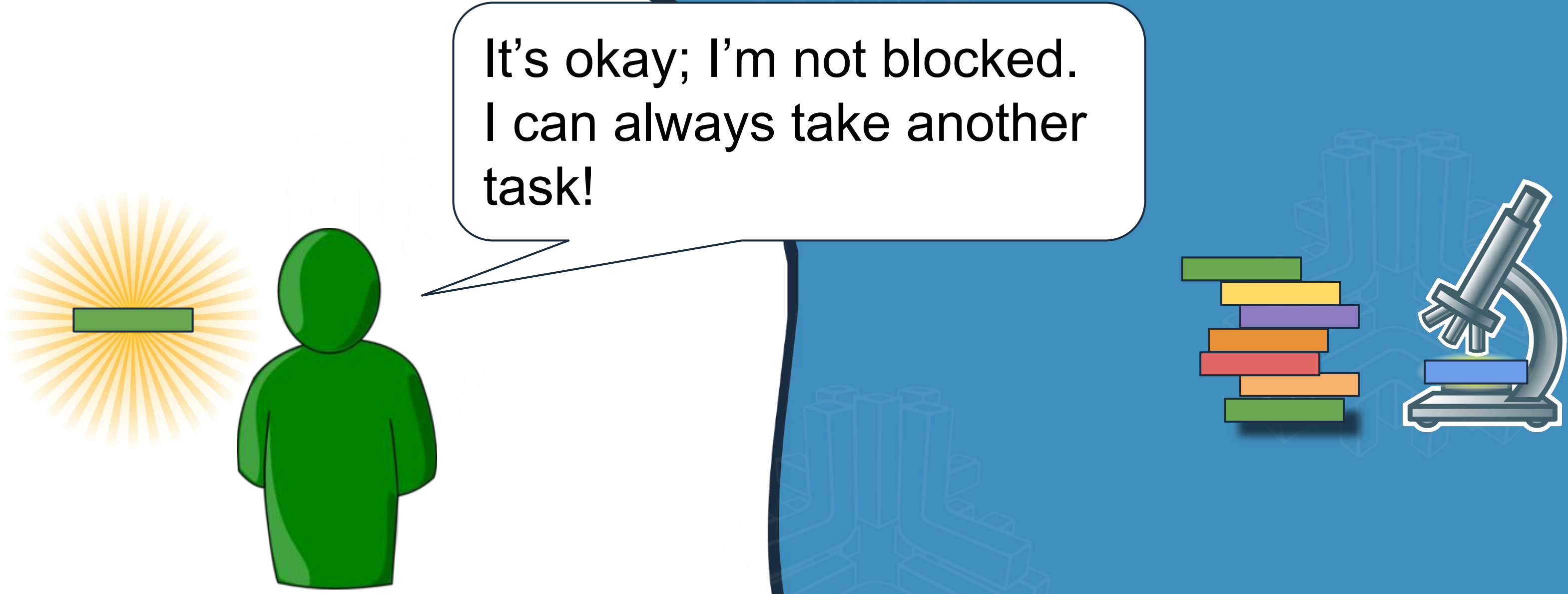
Each person is imperfect.

Any change may have defects.

It takes time for code to cycle between people and departments in most organizations.



Each rejection from QA slows a feature by days or weeks. Some code comes back more than once.

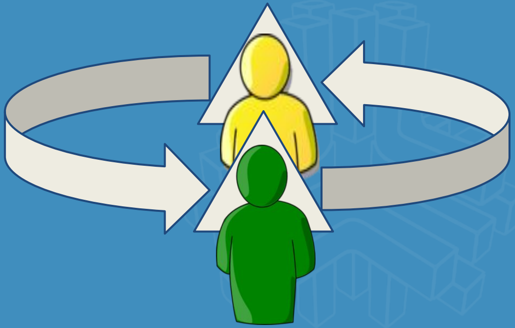

It’s okay; I’m not blocked. I can always take another task!

I've read that the industry average queue time for a PR is 5 business days.

I’ve no data on average time to turn around after a rejection

We’ve seen it take 2 months or more.

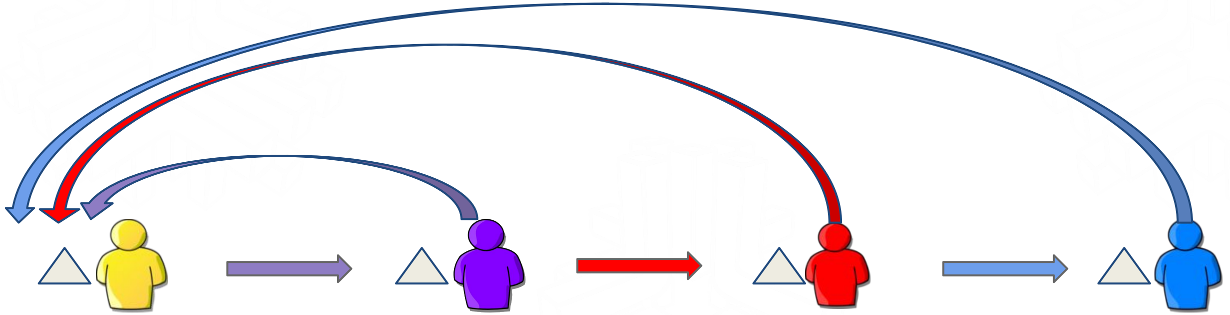
No fair!
I’m already on another ticket!
N Days
to correct



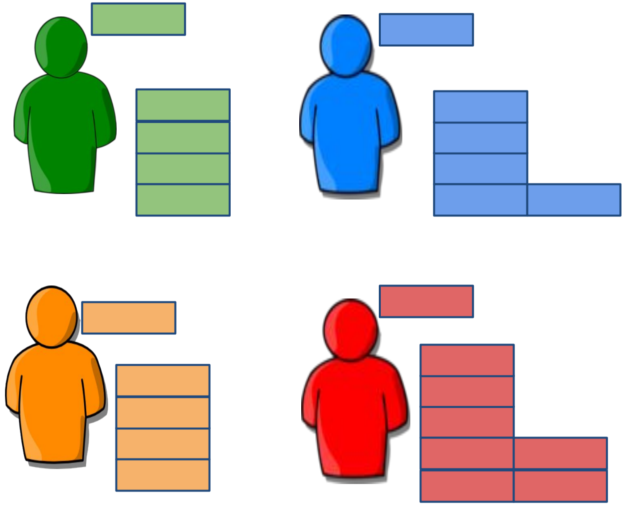
How many potential cycles do you have in your process?

industrial logic

How many NESTED loops in your process?

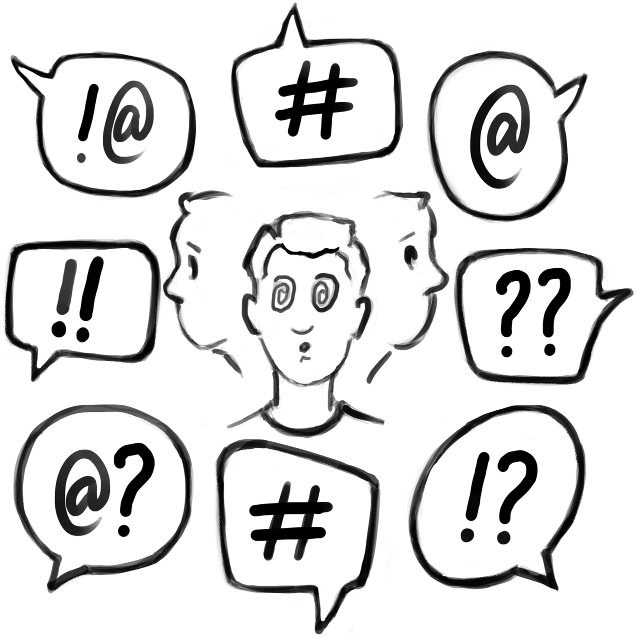


Work has to repeat each inner loop in order to make another pass at any outer loop.



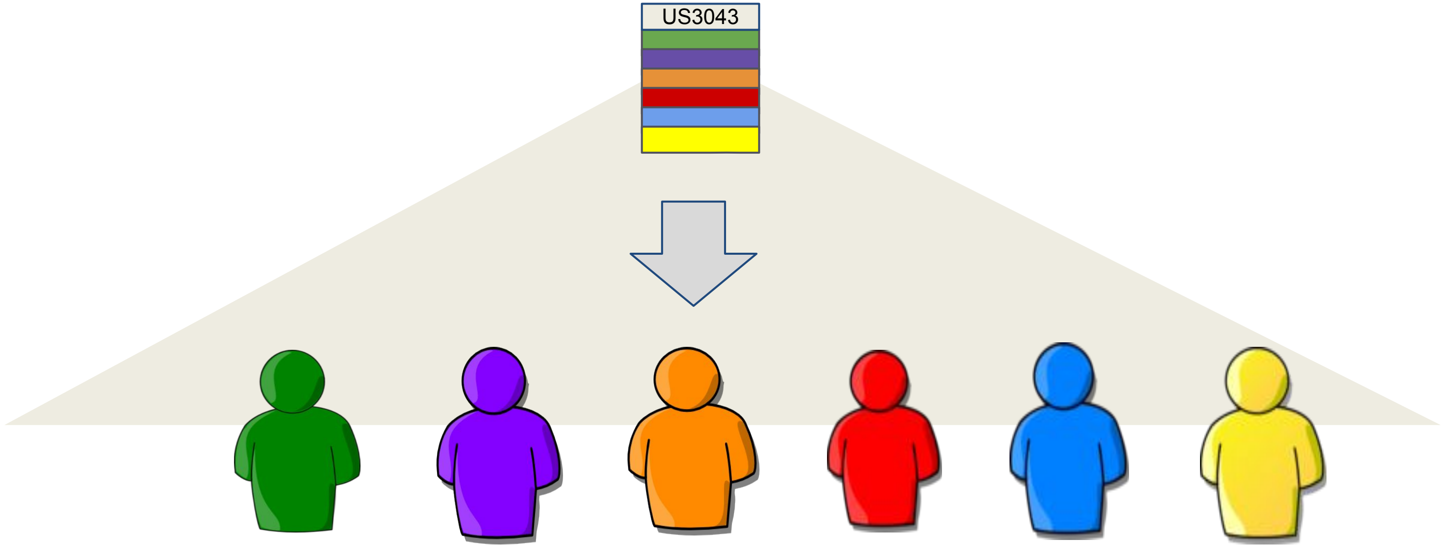
How much work isn't truly in progress?

How much partially-done work is waiting in queues?



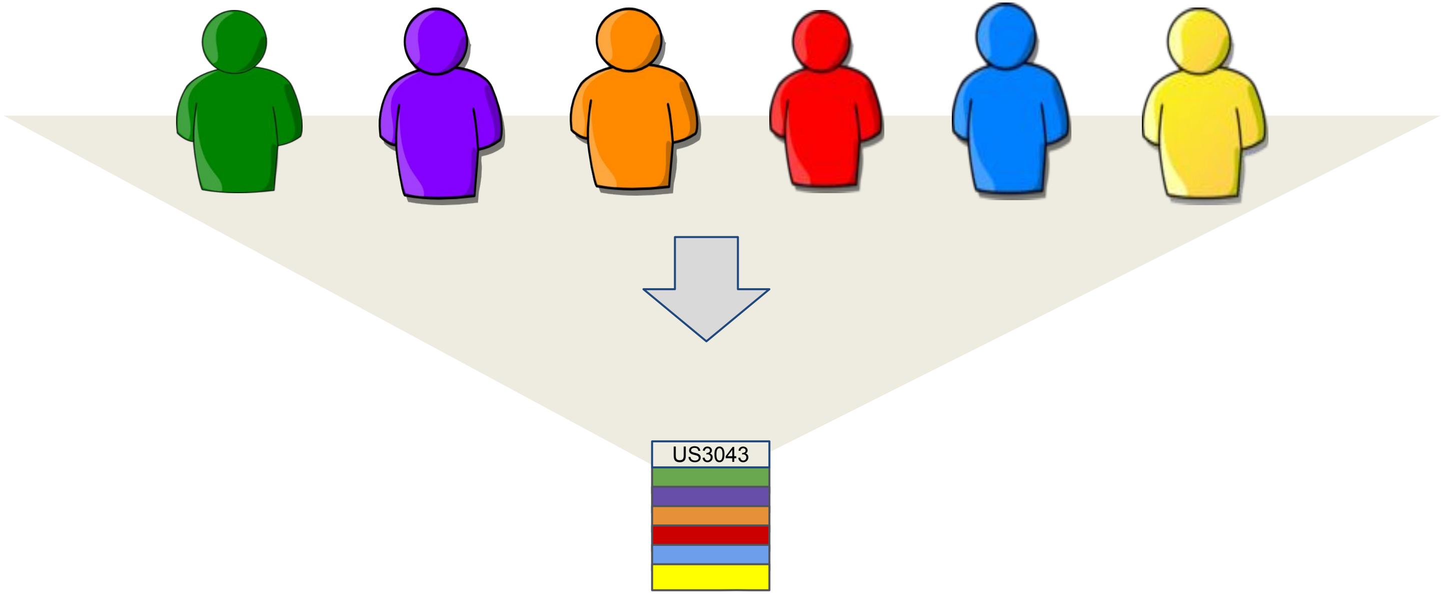
Each work item associated with your name is an invitation to interruption.

Work Scatter



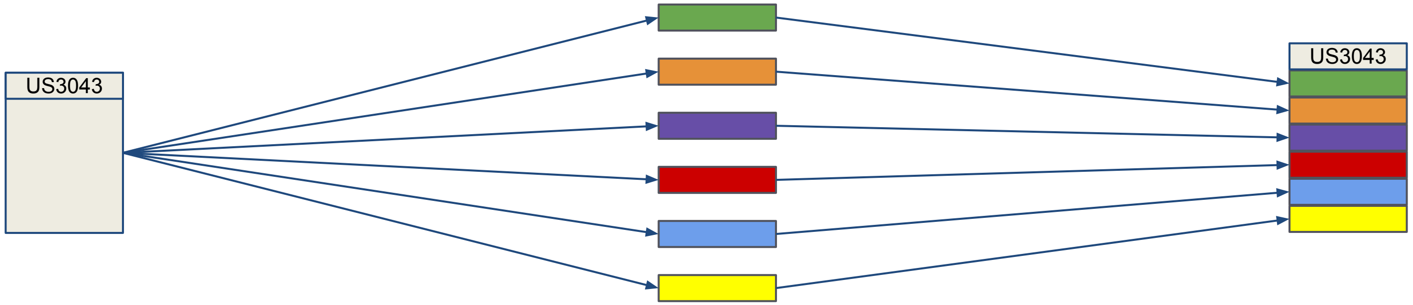
Splitting & assigning work per person’s skills is senior-level work requiring considerable design up front.

Gather



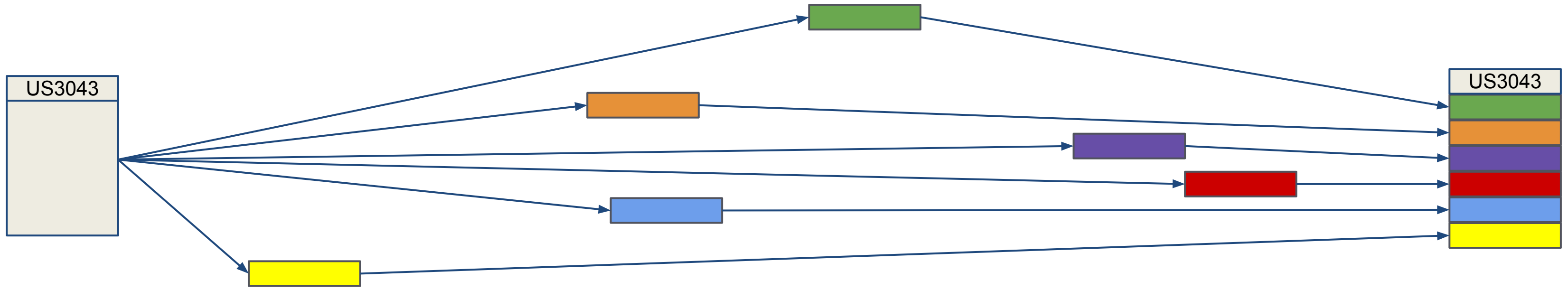
Ideally, the work will fit together w/o error when integrated.

Intention of Scatter/Gather



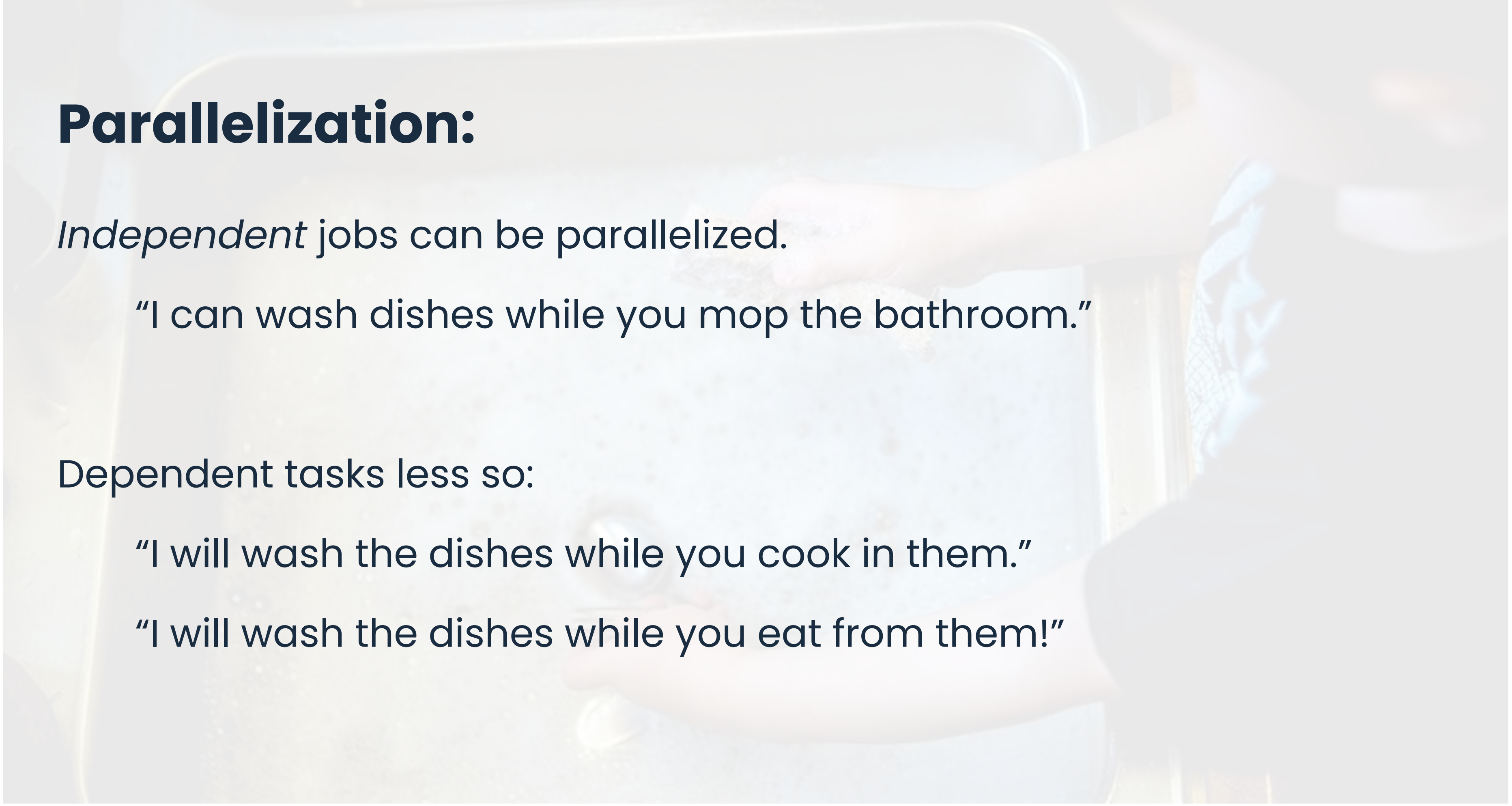
Because the work can be done in parallel, it should be done sooner than if the work, divided into the *same individual parts*, was done in a serialized manner.

Reality of Scatter/Gather



Work is seldom done in parallel…

… nor can it be.



Parallelization:

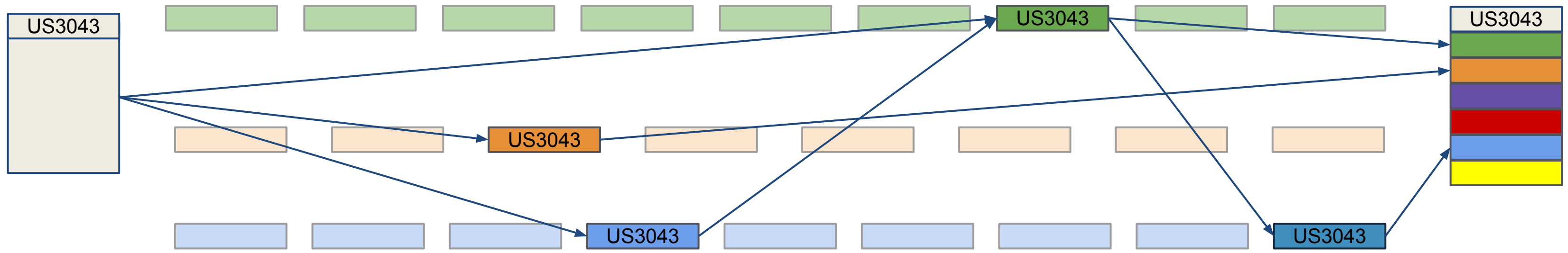
Independent jobs can be parallelized.

“I can wash dishes while you mop the bathroom.”

Dependent tasks less so:

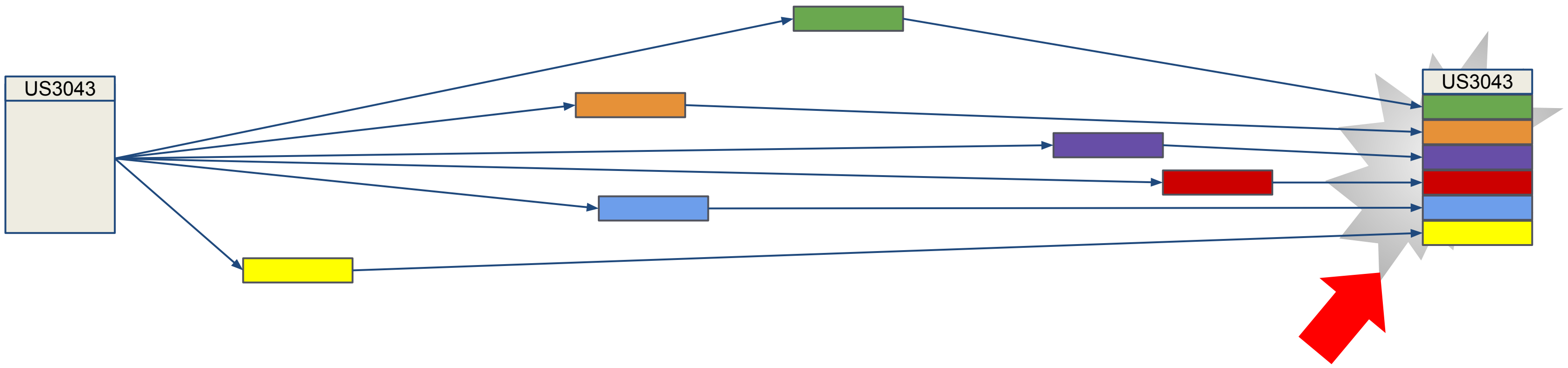
“I will wash the dishes while you cook in them.”

“I will wash the dishes while you eat from them!”

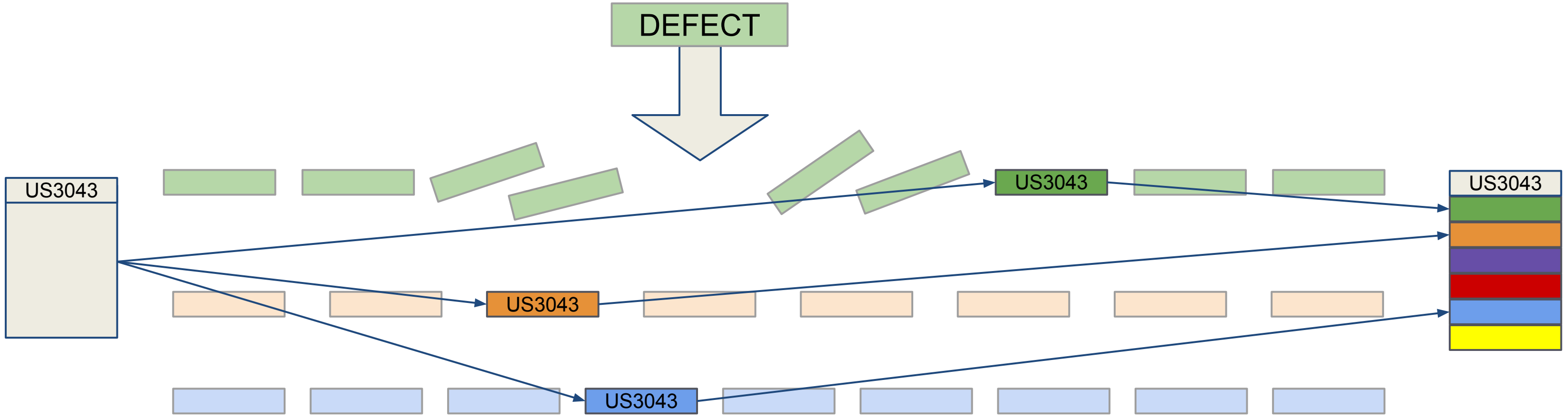


Each contributor has their own work queue and priorities.

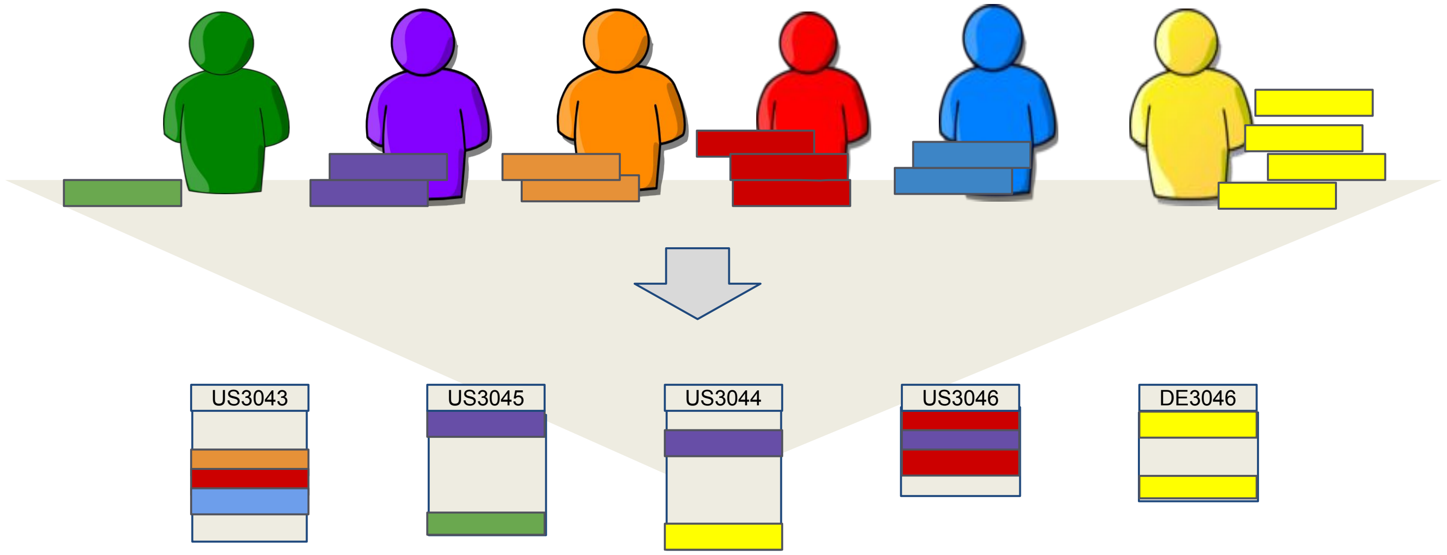
Tasks are not all independent

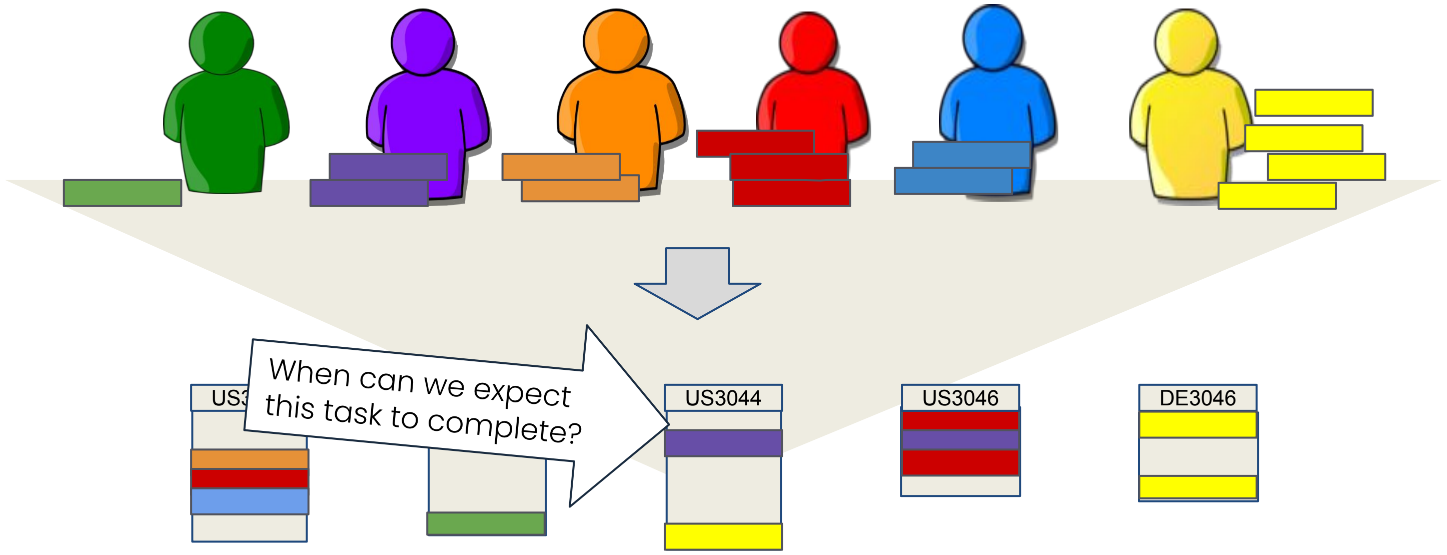


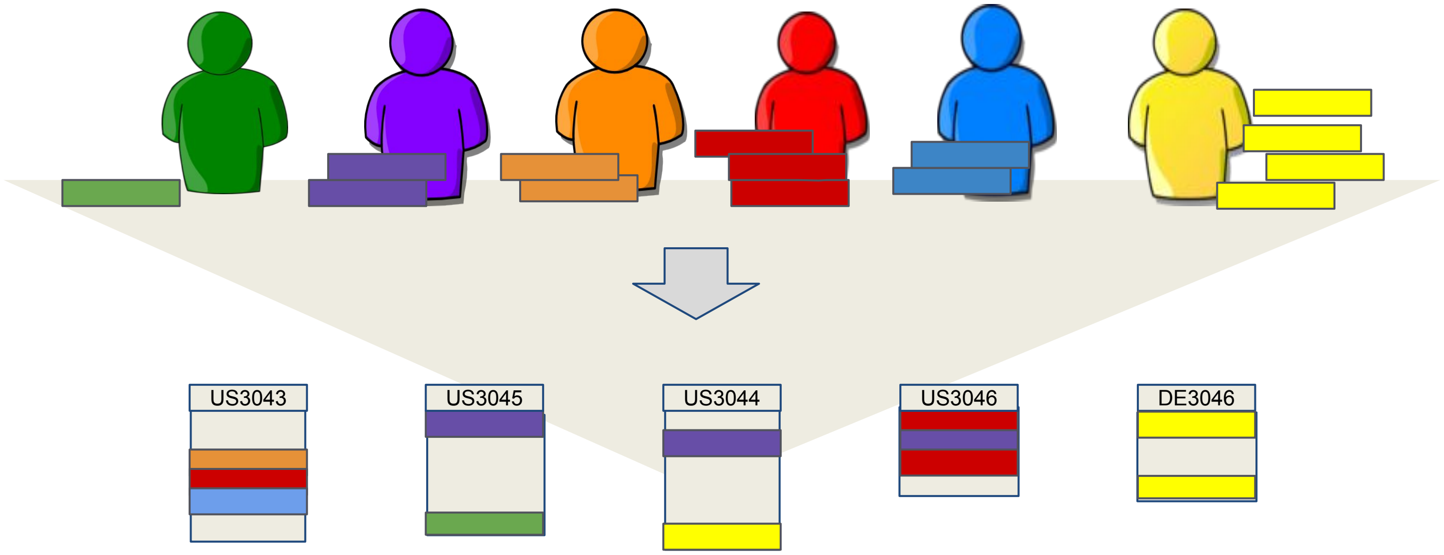
Because pieces were worked on by different people at different times, late integration failures are likely to loop defects back.



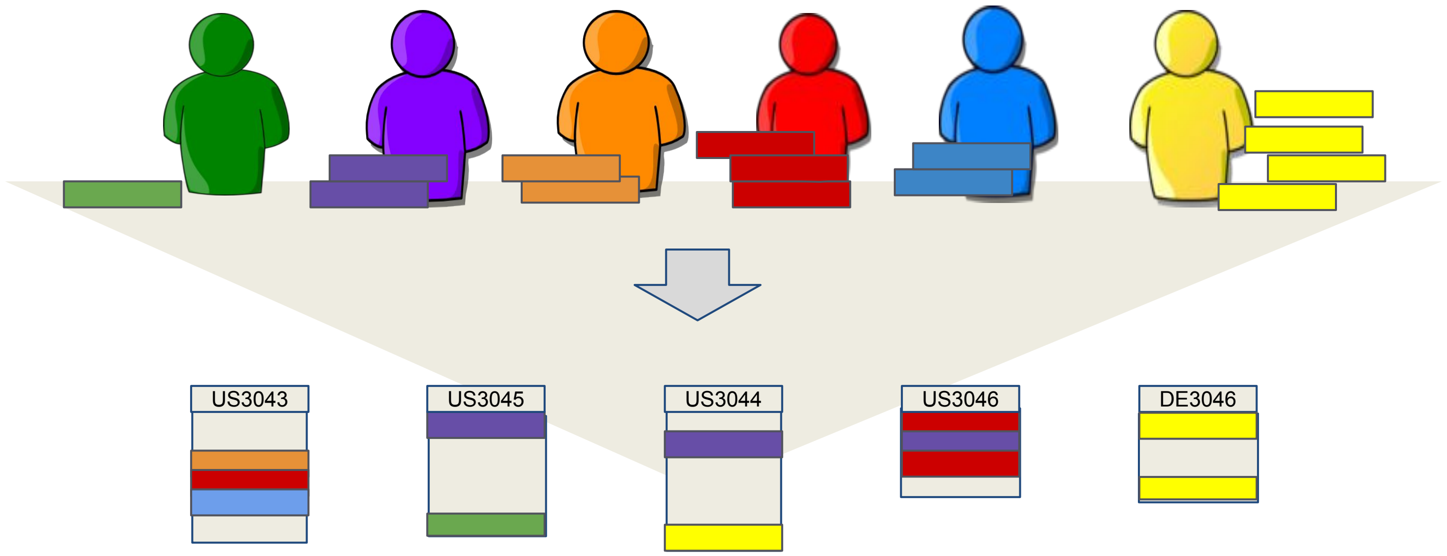
Each defect that loops back will disrupt a contributor’s queue of work, delaying their work on any other features.


US3043
US3045
US3044
US3046
DE3046


US3
US3044
US3046
DE3046
When can we expect this task to complete?



With *many* stories split, scattered, and active at the same time, predictability is lost.



Is this why we need sophisticated electronic systems: to help us track status of multiple, dependent, scattered work items?

Home > Blog > Faster and More Predictable

Faster and More Predictable

Tim Ottinger

03 Dec 2021 - 25 min read

facebook twitter linkedin

Agile Culture Mob Programming Pair Programming Process Improvement

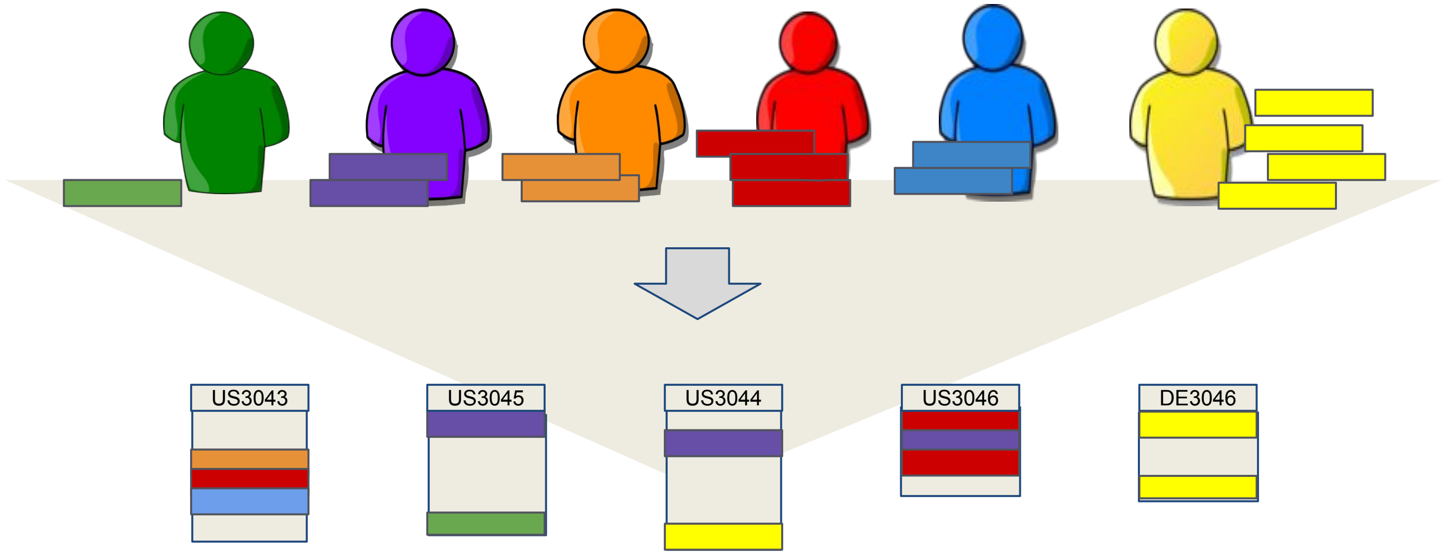
Everyone wants their software development organization to be faster and more predictable.

For most organizations, this is possible.

Tim Ottinger

Senior Consultant
Round Lake Beach, IL - USA

Contact

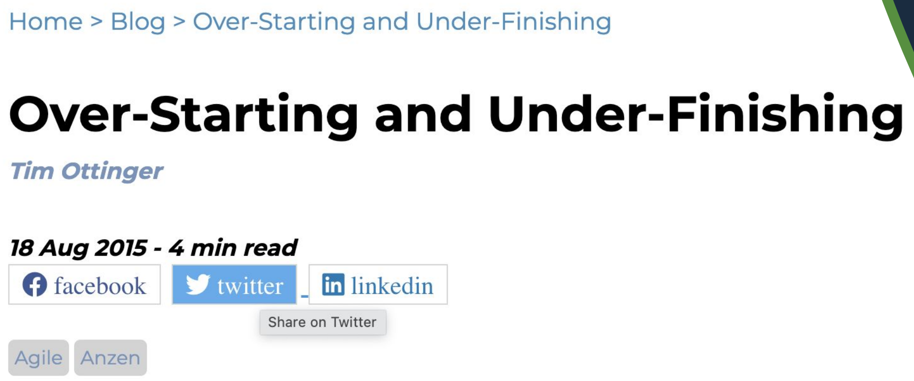


Do we receive higher throughput for the loss in predictability?

HOW 'BOUT NO

We would like to think so.

We take it on faith that we are getting more done since we are working on so many things and everyone is so busy!



Home > Blog > Over-Starting and Under-Finishing

Over-Starting and Under-Finishing

Tim Ottinger

18 Aug 2015 - 4 min read

facebook twitter linkedin

Share on Twitter

Agile Anzen

industrial logic

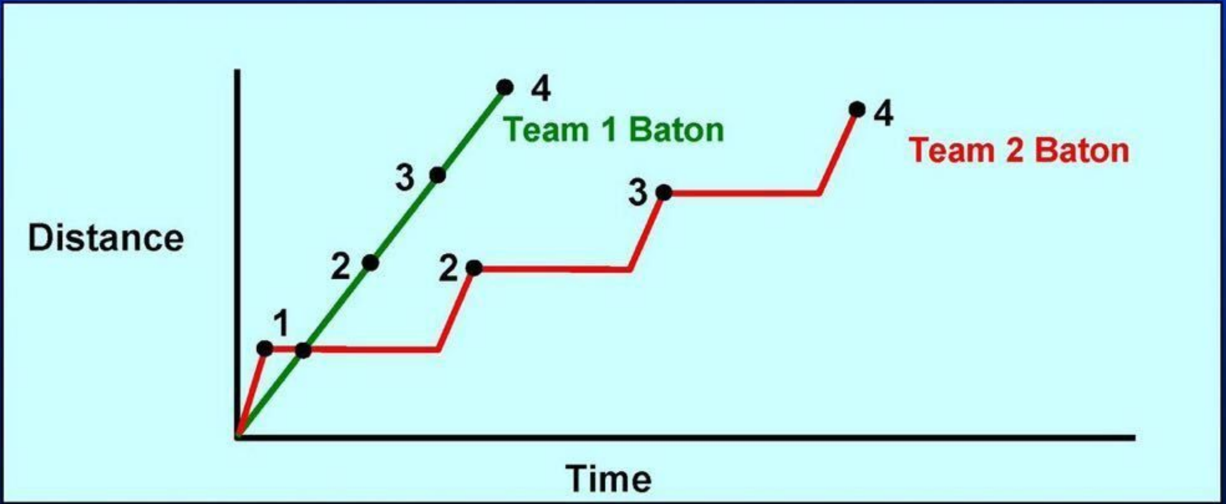
WHEW: we made it through the dark spaces.

Here are some kittens.

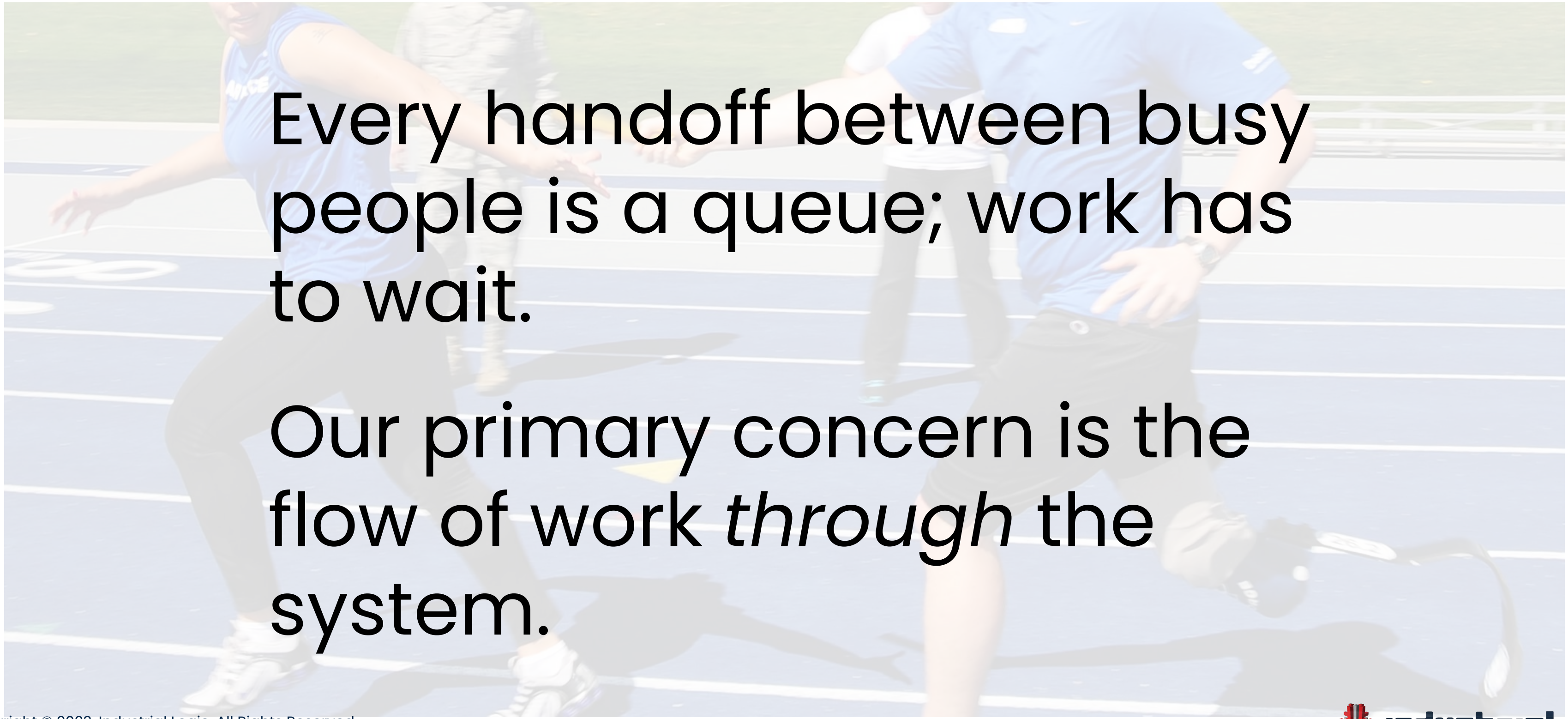
Watch the Baton, not the Runner!

Runner

Baton



Watch the Baton, Not The Runners

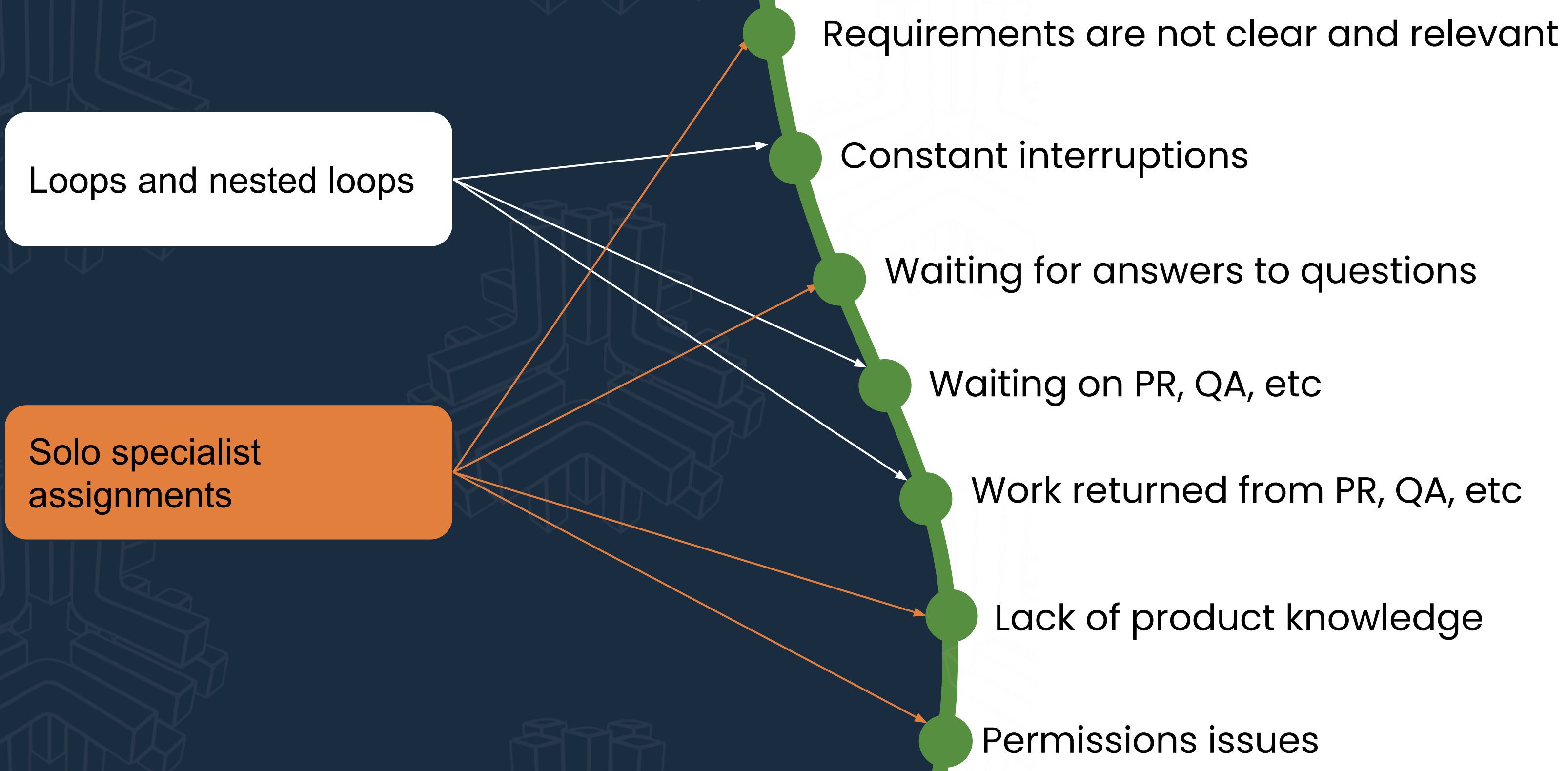


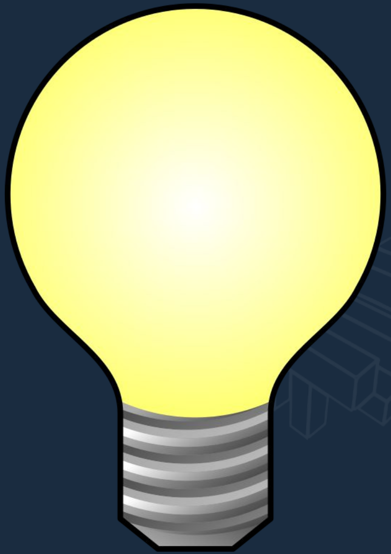
Every handoff between busy people is a queue; work has to wait.

Our primary concern is the flow of work *through* the system.

Oh, hey!
Aren't these mostly consequences of scatter-gather and solo work assignments?

- Requirements are not clear and relevant
- Constant interruptions
- Waiting for answers to questions
- Waiting on PR, QA, etc
- Work returned from PR, QA, etc
- Lack of product knowledge
- Permissions issues


Loops and nested loops
Solo specialist assignments
Requirements are not clear and relevant
Constant interruptions
Waiting for answers to questions
Waiting on PR, QA, etc
Work returned from PR, QA, etc
Lack of product knowledge
Permissions issues



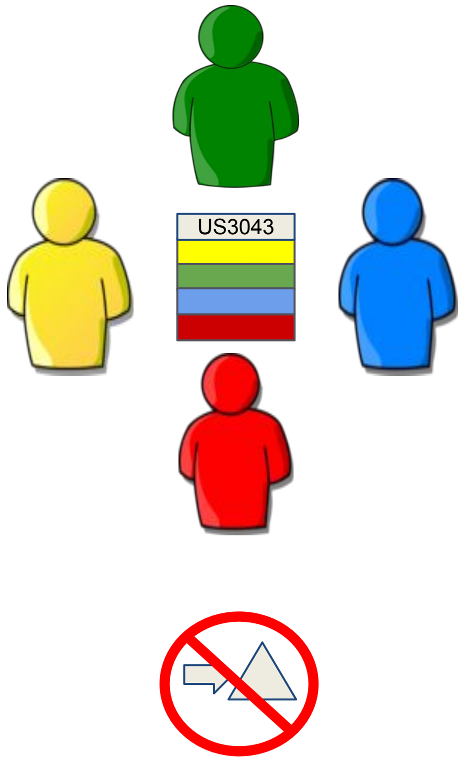
What if...

Instead of *splitting the work*,

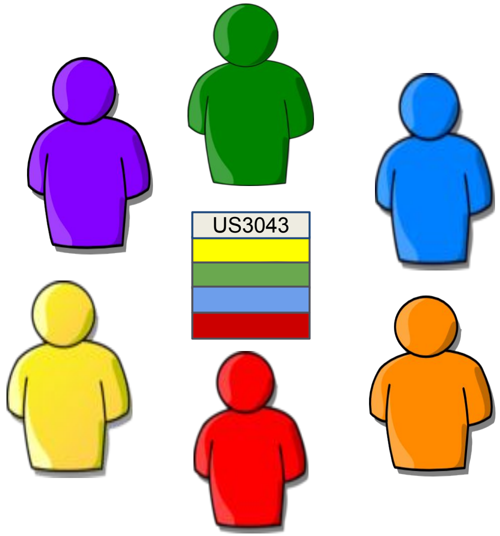
we *gathered the people*?

Gather people to avoid queues and handoffs between them.

Start together
Work together
Finish together



industrial logic



Include those who might otherwise reject/loop work back.

Start together
Work together
Finish together

For the scrum people in the room…

Scrum Teams are cross-functional, meaning the members have all the skills necessary to create value each Sprint. They are also self-managing, meaning they internally decide who does what, when, and how.

The Scrum Guide, 2020

For the XP people in the room…

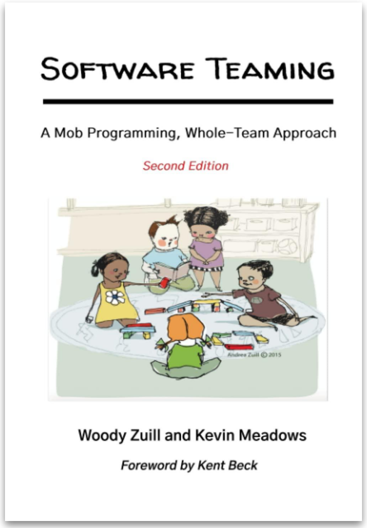
*XP teams are **self-organizing and cross-functional**. This has two important consequences: first, they're responsible for their own success. This means teams define success (by interviewing stakeholders and sponsors), create plans to achieve success, and execute on those plans without explicit management direction.*

*Second, **XP teams include all the expertise necessary** to do so.*

*In practice, XP teams are composed of business experts ("customers"), implementation experts ("programmers"), and quality experts ("testers"). T**he whole team works together** to create its own plans and deliver successful software. No single person is "in charge." Instead, leadership shifts fluidly with the situation.*

James Shore

It Has Always Been About Working Together



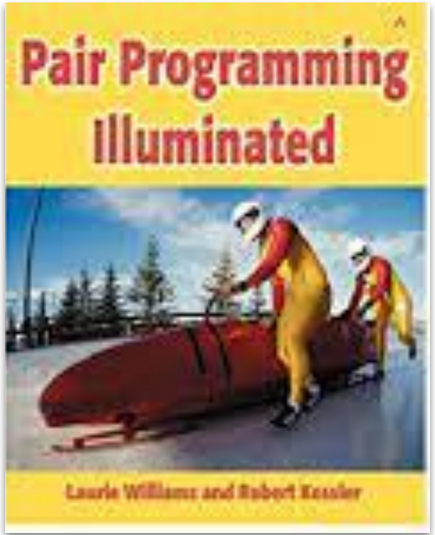
Software Teaming

A Mob Programming, Whole–Team Approach

Second Edition

Woody Zuill and Kevin Meadows

Foreword by Kent Beck



Pair Programming Illuminated

Laurie Williams and Robert Kessler

Unevenness of skill and product knowledge

Each important skill need only be in the team, rather than in each individual.

Teams can review the code as they are writing it, from multiple viewpoints.

Code is vetted and tested before it is committed, eliminating queues.

industrial logic

Lower WIP

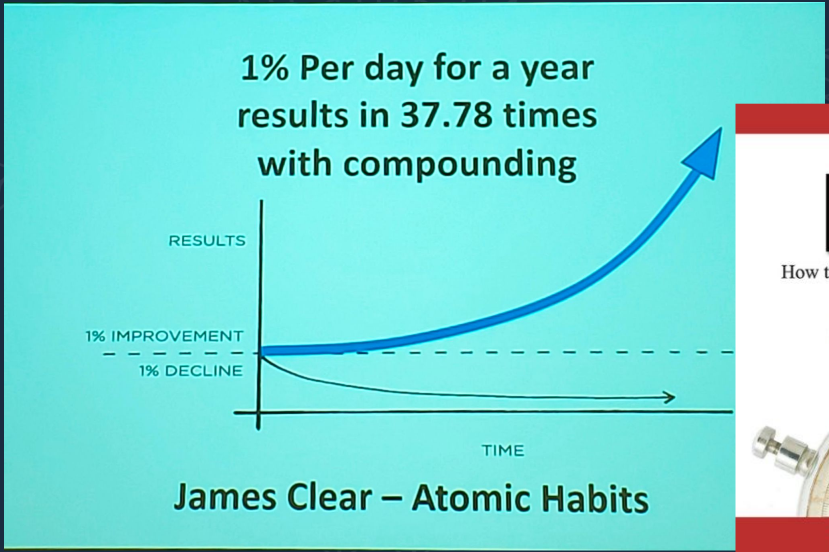
With fewer things in progress there is less to keep track of.

Work is sliced to deliver, not to developer's specific skill sets.

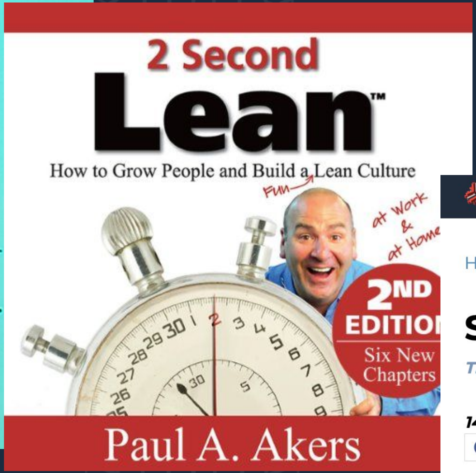
By Tim Ottinger

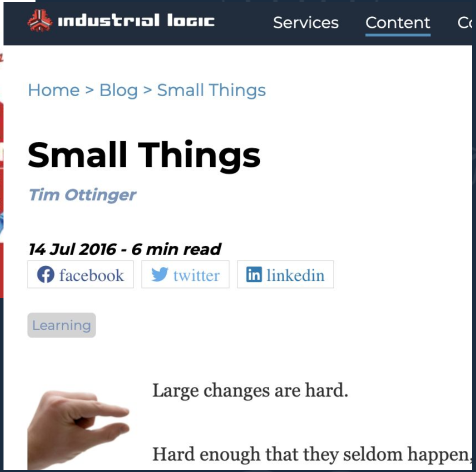
Pitfalls Of Solo Work

Is dividing work among individuals really effective? Maybe not...


1% Per day for a year results in 37.78 times with compounding

James Clear – Atomic Habits





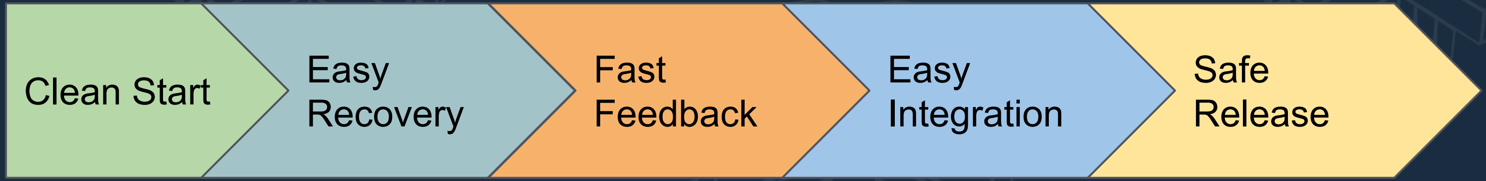
Small Things

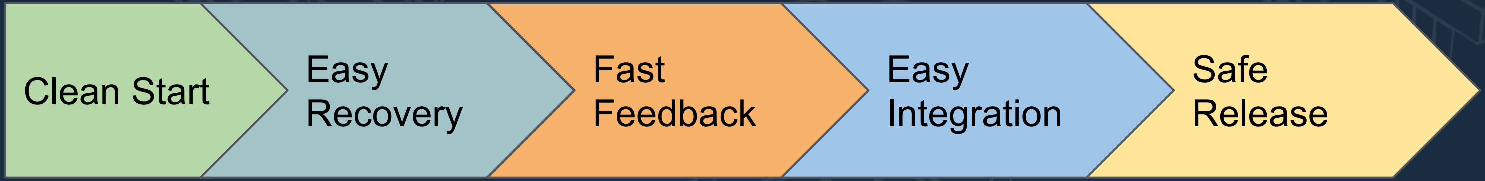
Tim Ottinger

14 Jul 2016 - 6 min read

Large changes are hard.

Hard enough that they seldom happen,


Clean Start
Easy Recovery
Fast Feedback
Easy Integration
Safe Release

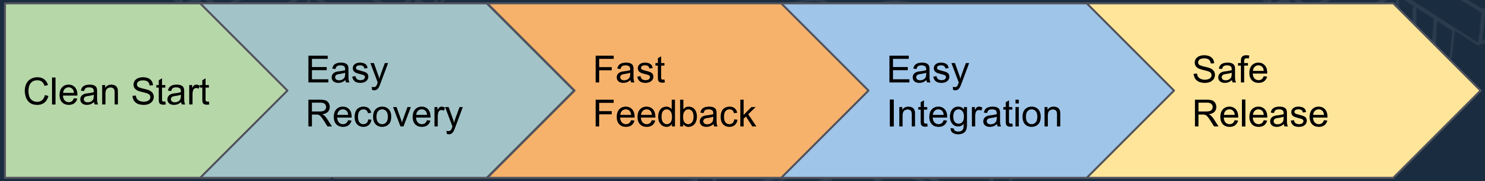


Check git status, it should be 100% clean (no leftovers)

Check out from the main branch

All Tests run GREEN

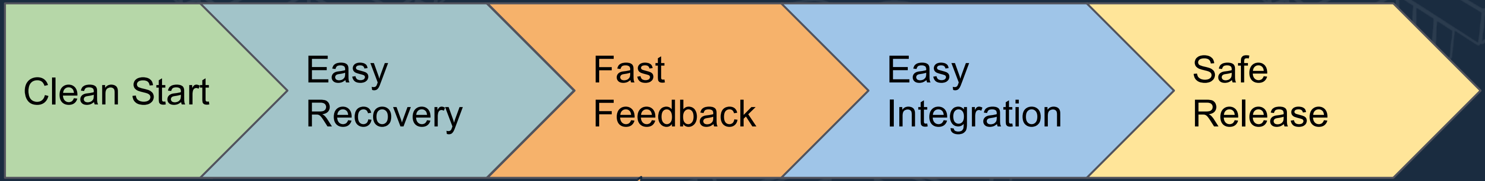
You know that you have no unfinished work



Make small changes

Work with tests so that you can spot an error quickly

Commit frequently so you can safely roll back

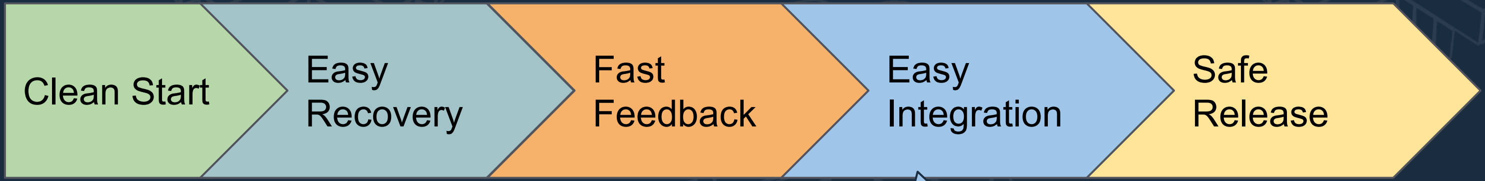


Rely on tests as you work

The whole team deliberates together

Integrate frequently to spot incompatibilities

Sponsors review product as it is being developed

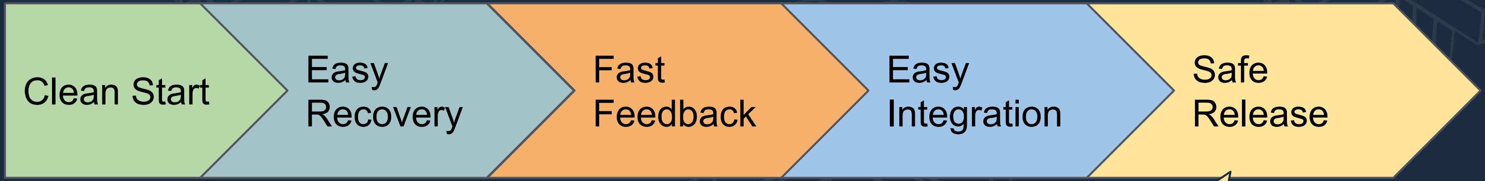


Frequently pull from main so you spot issues early

Test before and after integrating from main

Test before pushing to main

Monitor the CI pipeline

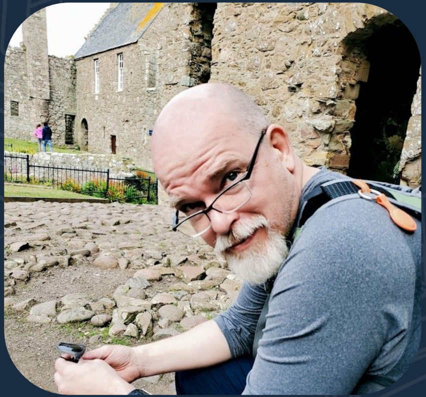


Automate releases so no human error is introduced

Create zero-downtime releases

Automated tests cover key functionality

Feature flags/toggles used where necessary

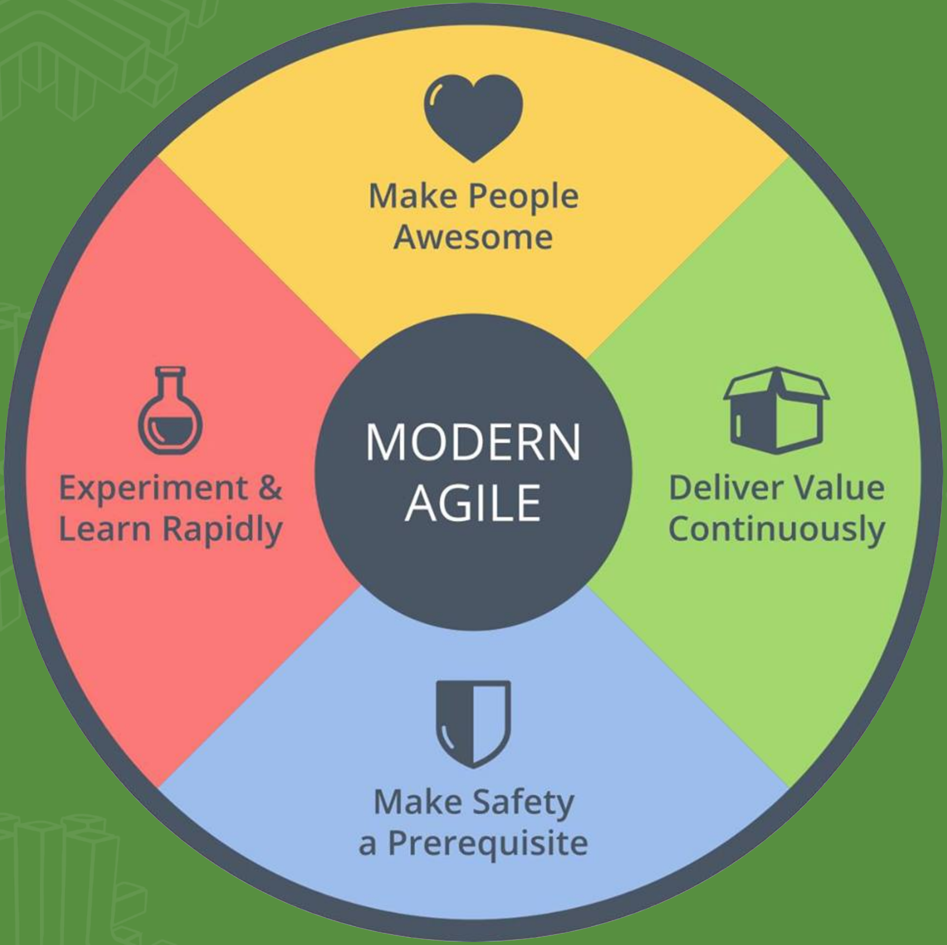


I'm moving to Scotland!

I will serve Europe, Asia, and the US from my new home in Edinburgh.

Availability starting in October.

Check out our public workshops!


Make People Awesome
Experiment & Learn Rapidly
MODERN AGILE
Deliver Value Continuously
Make Safety a Prerequisite