
ThinkingLabs
2021 Ghent lightening festival
Kunsthal Gent
Lange Steenstraat
Non-Blocking Continuous Code Reviews
Thoughtful Thierry
KUNSTHAL GENT
in/tdpauw
@tdpauw.bsky.social
@tdpauw@mastodon.social
thinkinglabs.io

Why are we doing Code Reviews? 🤔

But few teams have made a conscious decision about what they want to achieve with their code reviews.

– Pia Fåk Sunnanbo, I Hate Pull Requests

My reasons ...

- Mentoring, learning
- Knowledge sharing
- Check on readability and maintainability
- ~~Maintain consistency in design~~

(*that was a mistake, should have been a design workshop*)

Other reasons ...

4-eyes principle

Killer argument to slow everything down
and drive down quality.

Pair or Team Programming
are more effective

“Sense” of security.

I don’t trust my code.

Dilution of responsibility.

-> won’t be fixed by a process

What Code Reviews should not be?

- Exercise power
- By seniors missing mentoring skills
- Blaming or harming
- A miserable experience
- Gatekeeping
- A must

Non-Blocking
Continuous Code Reviews

A Team's Tale

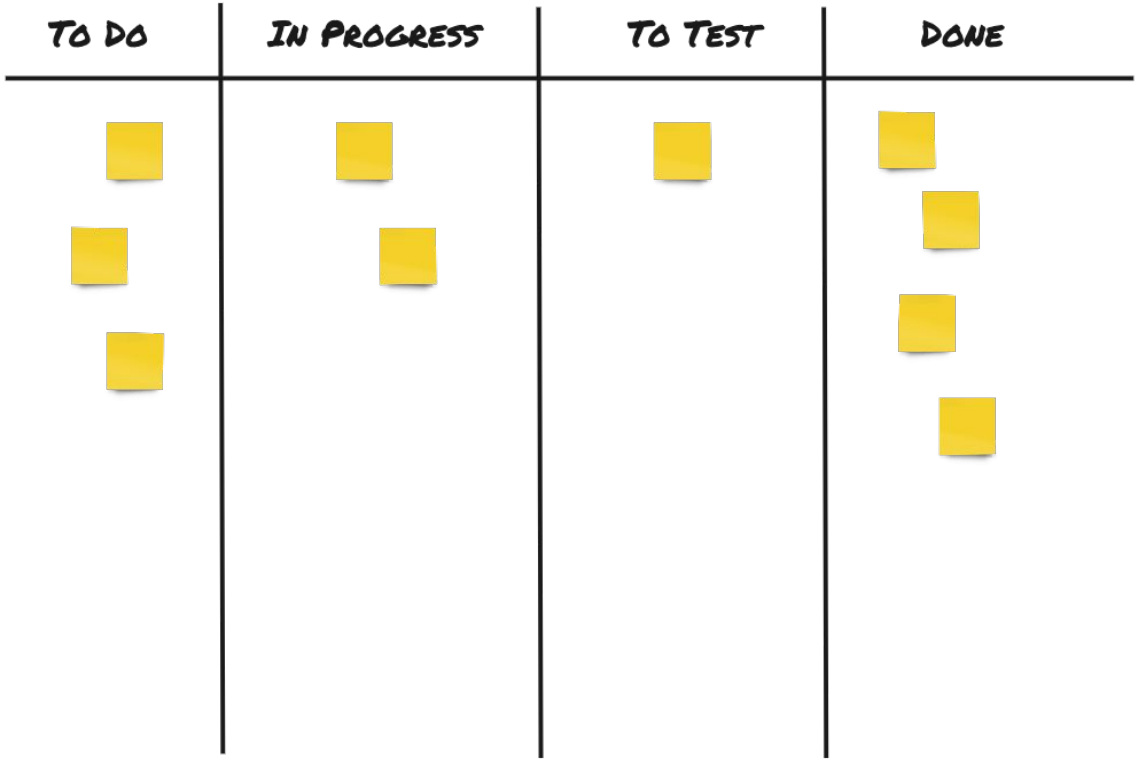

ThinkingLabs
Ghent, Belgium
On the left, Korenlei
On the right, Graslei
Feature Branching is Evil
Tenderhearted Thierry
takes questions at the end of the session
@tdpauw@mastodon.social
thinkinglabs.io

ThinkingLabs

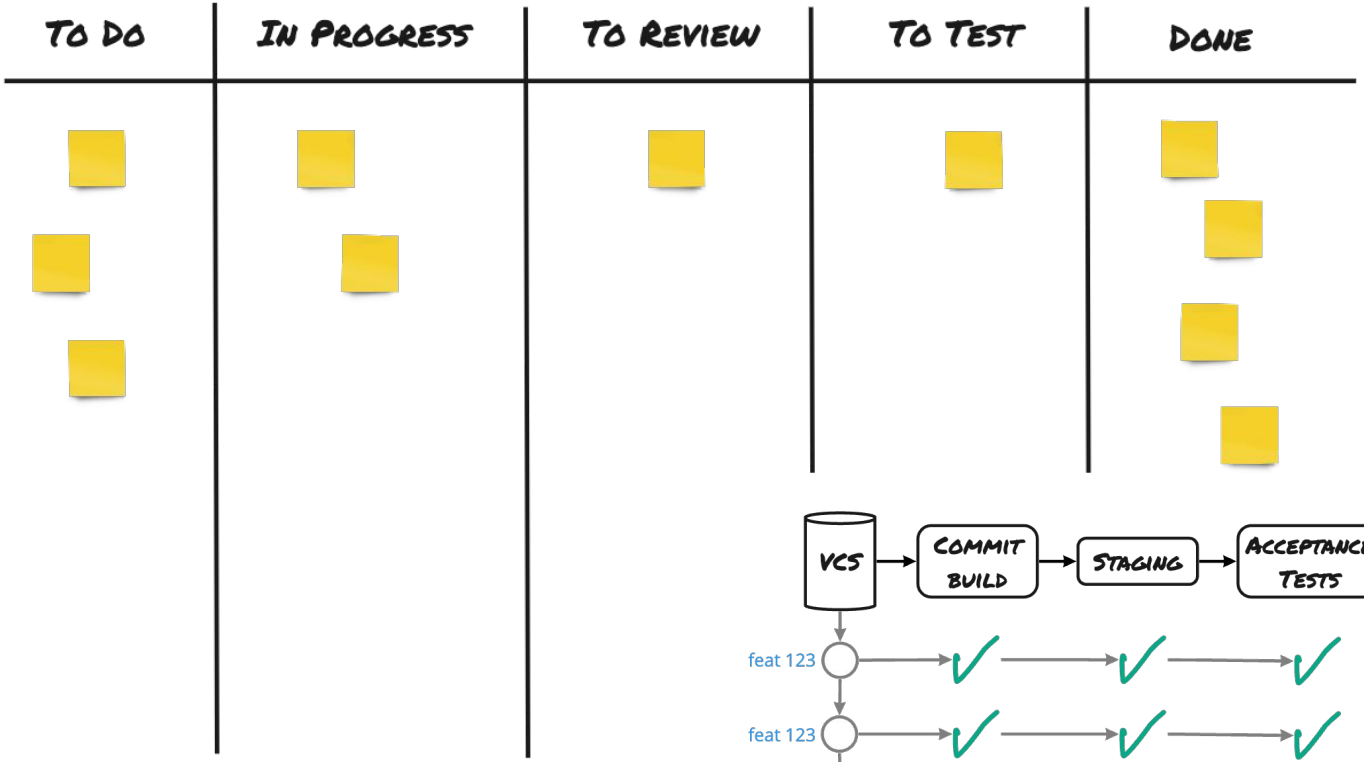
Two key ideas ...

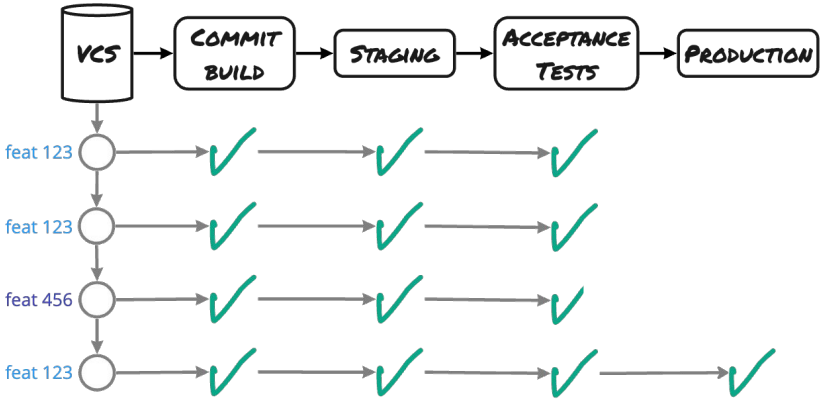
Establish a continuous review process.

ThinkingLabs



ThinkingLabs

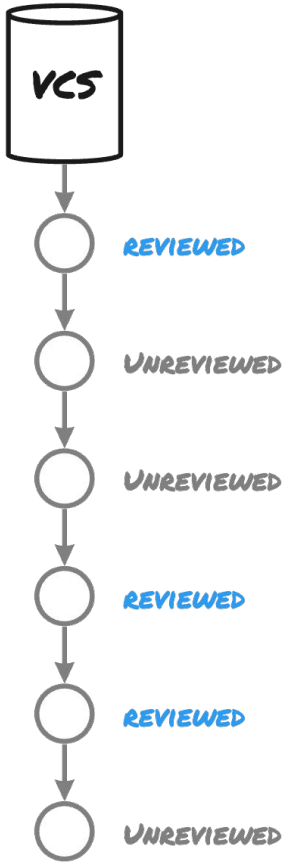




ThinkingLabs

If we commit changes
directly to the remote mainline,

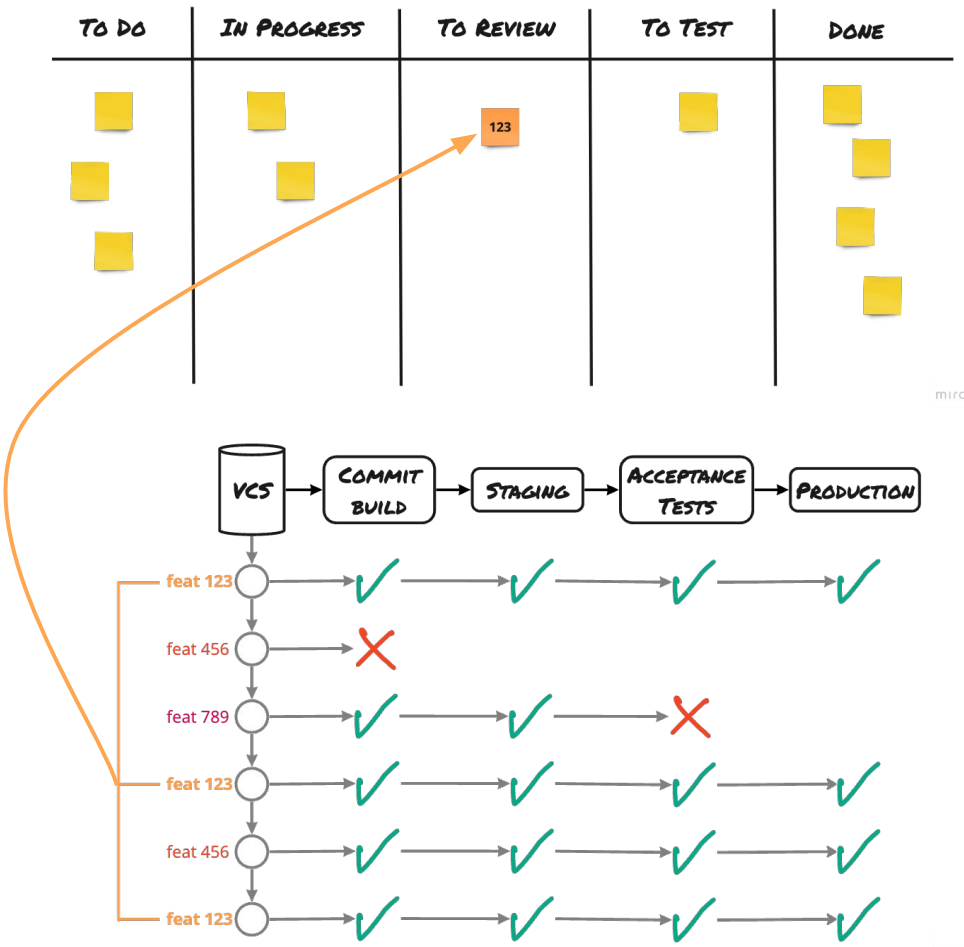
we have unreviewed code
alongside reviewed code.



ThinkingLabs

Second key idea ...

Stop the code review being definitive for release.



Unreviewed code could end up in production, whether the feature was finished or not.

Unreviewed code ≠ a bug !!

=> automated and exploratory tests
catch bugs

The unreviewed code is code that works.
It passed all tests!

ThinkingLabs

Benefits

Transaction Cost

the cost of sending a batch
to the next stage

It it takes 10 min to make a change, the change is blocked until it is reviewed and it takes 2 hours to get a review ...

92.3%
wait time 😳

High cost of code review!

=> ask less often for a review 🤷

@tdpauw@mastodon.social
@tdpauw.bsky.social

Expensive process

works against making small changes

-> **less refactoring**

-> **less incremental development**

the wait time incentivises Work in Progress

ThinkingLabs

Little's Law

the higher the WIP
the more delay

$$\text{Lead time} = \frac{\text{WIP}}{\text{Throughput}}$$

To get something out sooner ...
need to reduce the transaction cost!

Minimise the cost of code review.

ThinkingLabs

Benefits (bis)

Transaction Cost is nearly zero.

=> **work in small increments**

With Non-Blocking Code Reviews
we never slow down,
changes are **never blocked for delivery**.

No gating!

With Non-Blocking Reviews
we have **early, fast feedback**.

With Non-Blocking Code Reviews
no new work is started before finishing.

=> less Work in Progress

=> because Little’s Law: less delays

With Non-Blocking Code Reviews
no pressure to improve code quality.

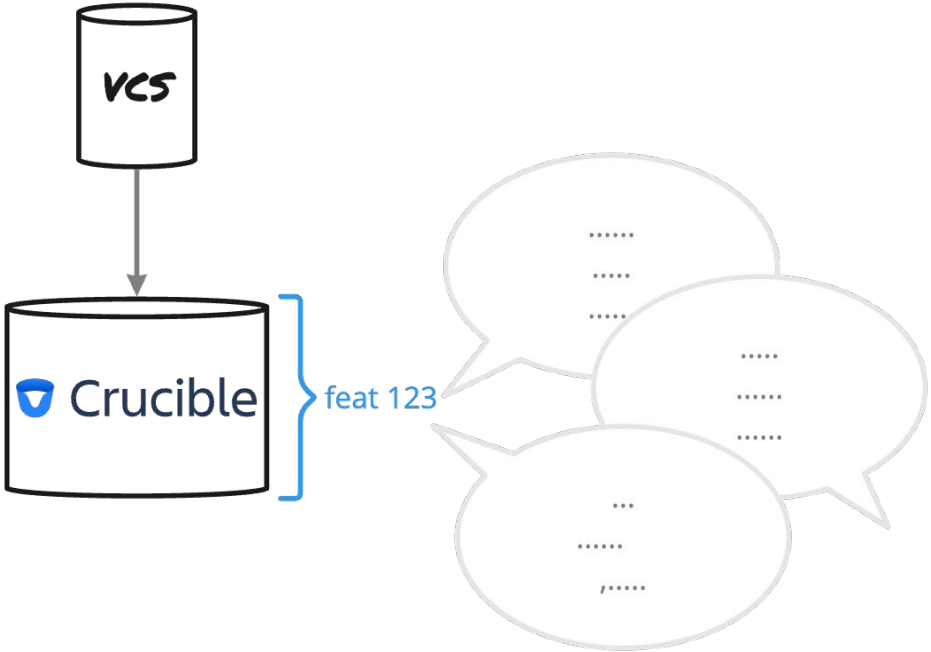
Nobody is waiting.

With Non-Blocking Code Reviews
there is **high trust**.

Tooling

ThinkingLabs

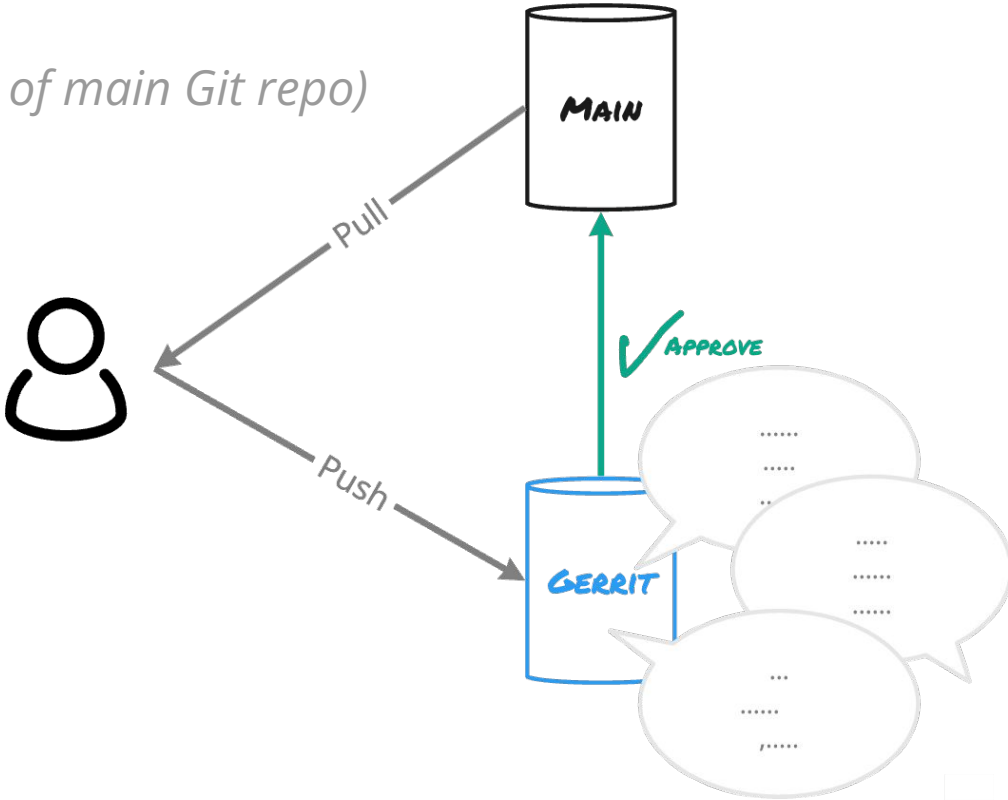
- We used: Atlassian Crucible *(discontinued)*



- JetBrains UpSource *(discontinued)*
- Facebook Fabricator *(discontinued)*

ThinkingLabs

- Gerrit *(review Git repo in front of main Git repo)*



- Auctionet Ex-remit *(barsoom/ex-remit on GitHub)*
- ReviewBoard *(similar to Crucible)*

- Or … no tooling at all 🤷

Gotcha with Non-Blocking Code Reviews

It does not work for regulated industries.

=> Pair or Team Programming
is the better solution

Conclusion

Pair or Team Programming
are still a superior delivery option.

But can be a cultural stretch.

Non-Blocking Review strategies are **significantly better** than any of the alternatives.

- No gating
- Deliver faster without delays
- Less WIP
- No context switching
- No urgency to fix quality issues
- Features grow incrementally, commit by commit
- Encourages refactoring

ThinkingLabs

Hello, I am Thierry de Pauw

Acknowledgments:

Stefan De Moerloose (https://linkedin.com/in/stefandemoerloose/) for having been that amazing team manager open to try out all our crazy ideas.

Dave Farley (@davefarley77) for nudging me to write this down.

Giovanni Asproni (@gasproni@mastodon.social) for poking me to turn this into a positive story.

Lanette Creamer, Wouter Lagerwije, Diane Gombart, Falk Kühnel and Steve Berczuk for reviewing the slide deck.

Laphroaig and Bunnahabhain for the creative support.

The Article:
https://thinkinglabs.io/articles/2023/05/02/non-blocking-continuous-code-reviews-a-case-study.html

in/tdpauw
@tdpauw.bsky.social
@tdpauw@mastodon.social

Resources

Optimizing the Software development process for continuous integration and flow of work, Martin Mortensen

The Article: Non-Blocking Continuous Code Reviews, a Case Study, Thierry de Pauw

Code Complete, Steve McConnell

Facts and Fallacies of Software Engineering, Robert Glass

Joel on Software, Joel Spolsky

What is the purpose of Code Reviews, Thierry de Pauw, a discussion on LinkedIn

I've found something better than PRs, the video from Dave Farley on the topic

From Async Code Reviews to Co-Creation Patterns, Dragan Stepanović

I Hate Pull Requests, Pia Fåk Sunnanbo

Problems with Pull Requests and How to Fix Them, Gregory Szorc

On the Evilness of Feature Branching - But Compliance, Thierry de Pauw

How to make sure tests are of quality, Thierry de Pauw, a discussion on LinkedIn