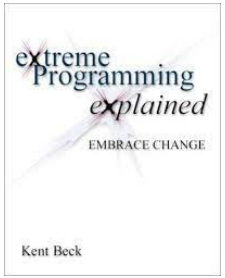
The Dusk of Manual Code Review on Pull Requests

@FreddyMallet
@codereviewpad
reviewpad.com

I’m about to turn 50

IN THE PAST



TODAY

170 M

Pull Requests Merged in 2021 on Github

Octoverse Report 2021
https://octoverse.github.com/writing-code-faster/

At the inception

Pull Request was a way to welcome **unexpected** contributions from **unknown** and **not trustworthy** developers

Today

Pull Request is the **backbone** of any **devops** infrastructure to safely inject **expected** code changes done by **trustworthy** developers.

Automation Everywhere

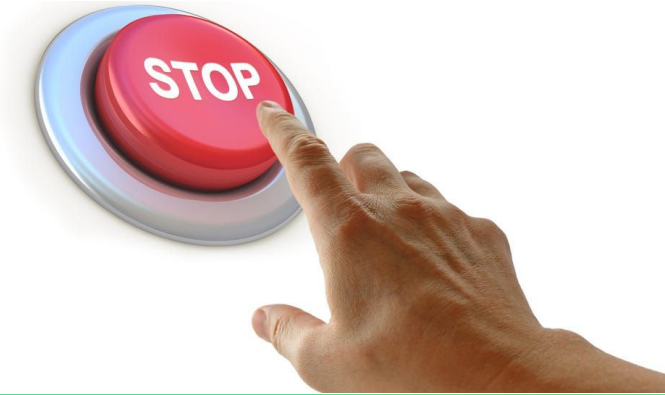
- To detect build failures
- To detect test failures
- To detect legal issues
- To detect obvious bugs
- To detect obvious vulnerabilities
- To detect compliance issues
- To deploy

But no automation will ever formerly prove the soundness of a code change

So we “Stop the Line” on each PR

The manual review step is

- Potentially **unsafe** and especially **at scale**
- More a **validation** step than a code review step
- A costly **transactional** step in time and energy



Since 2015: DORA Metrics

A real paradigm shift

https://cloud.google.com/blog/products/devops-sre/dora-2022-accelerate-state-of-devops-report-now-out

The focus is not anymore on the productivity but on the delivery performance

Software delivery performance metric	Low	Medium	High
Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	Between once per month and once every 6 months	Between once per week and once per month	On-demand (multiple deploys per day)
Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Between one month and six months	Between one week and one month	Between one day and one week
Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Between one week and one month	Between one day and one week	Less than one day
Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	46%-60%	16%-30%	0%-15%

Google 2022 Accelerate State of Devops Report
https://cloud.google.com/blog/products/devops-sre/dora-2022-accelerate-state-of-devops-report-now-out

Focus on Lead Time for Changes

	ELITE	STRONG	FAIR	NEEDS FOCUS
CODING TIME	< 12 hours	12 - 24 hours	24 - 38 hours	39 + hours
PICKUP TIME	< 7 hours	7 - 12 hours	12 - 18 hours	19 + hours
REVIEW TIME	< 6 hours	6 - 13 hours	13 - 28 hours	29 + hours
DEPLOY TIME	< 4 hours	4 - 48 hours	2 - 7 days	8 + days

LinearB Q1 2022 Labs study
https://linearb.io/engineering-benchmarks/

Based on a survey from more than **12’000** developers

“Detailed code reviews negatively affect software delivery performance”

Octoverse Report 2021
https://octoverse.github.com/writing-code-faster/

74%

percentage of PRs across all Github public projects without a single review comment from January to March 2020 *

Reviewpad: Looking Into 2020’S Pull Requests based on 2 631 366 pull requests across 283 034 projects
https://reviewpad.com/blog/2020s-pull-requests-part-ii/

64%

of Pull Requests are merged without any code changes as a result of the review *

Codeball 2022 study based on 2,340,078 PRs spread across 7,836 organizations
https://codeball.ai/blog/waiting-for-code-review

“With each additional reviewer, the chance to merge a PR in a day or less goes down by about 17%”

Octoverse Report 2021
https://octoverse.github.com/writing-code-faster/

SHIP / SHOW / ASK Model

By Rouan Wilsenach 2021
https://martinfowler.com/articles/ship-show-ask.html

The Principles

- SHIP
  Bypass the review when you know it's useless
  `approve, merge`
- SHOW
  Ask for a **non-blocking** review when you know your code is "good-to-go"
  `approve, merge, post-merge review`
- ASK
  Ask for a **blocking** review
  `assign`

Big Limitation

The **evaluation** is done by the author of the code change him/herself

What’s the difference between a non-blocking and a blocking review ?????

A change with low risk to impact the DORA Metrics

- Deployment Frequency
- Lead Time for Changes
- Time to Restore Service
- **Change Failure Rate**

What can be a risky code change ?

What can be a risky code change ?

- Pushed by a new joiner or new committer
- Involving an error prone instruction (regex, multithreading, ...)
- Containing an advanced design pattern
- Updating a permanent data structure (CB)
- Making a call to another system
- Done on a piece of code involved in some failures in the past
- Done on a piece of code surrounded by the @critical annotation
- Too big, too complex
- ...

Risk Assessment Model

To evaluate the probability of a code change to degrade the Change Failure Rate

We need a way to automatically assess the riskiness of a code change

Hold on! Isn’t it a bad practice to review a code after the merge?

Automation to the rescue

Main Features

- Configurable risk assessment rules
- Risk assessment rules automatically inferred from the past review activities
- Auto-approve mechanism
- Auto-merge mechanism
- Merge-Queue
- Configurable review assignment mechanism
- Risky code highlighter
- Configurable PR labelling mechanism
  (see kubernetes and reviewpad Github projects)

Specification and automation of the team's review process to decide **if** to review, **when** to review, **what** to review and **who** should review

Which are the Players?

Reviewpad
Codeball
Gitstream by Linearb
Mergify

reviewpad 

- Configurable risk assessment rules
- Risk assessment rules automatically inferred from the past
- Auto-approve mechanism
- Auto-merge mechanism
- Merge-Queue
- Configurable review assignment mechanism
- Risky code highlighter
- PR triage mechanism

CodeBall

- Configurable risk assessment rules
- Risk assessment rules automatically inferred from the past
- Auto-approve mechanism
- Auto-merge mechanism
- Merge-Queue
- Configurable review assignment mechanism
- Risky code highlighter
- PR triage mechanism

Mergify

- Configurable risk assessment rules
- Risk assessment rules automatically inferred from the past
- Auto-approve mechanism
- Auto-merge mechanism
- Merge-Queue
- Configurable review assignment mechanism
- Risky code highlighter
- PR triage mechanism

Gitstream by Linearb

Demo of reviewpad on reviewpad

Thanks !
Any Questions ?

@FreddyMallet
@codereviewpad
reviewpad.com