Making code review a more inclusive & productive collaboration

LeadDev San Francisco 2022
October 27, 2022

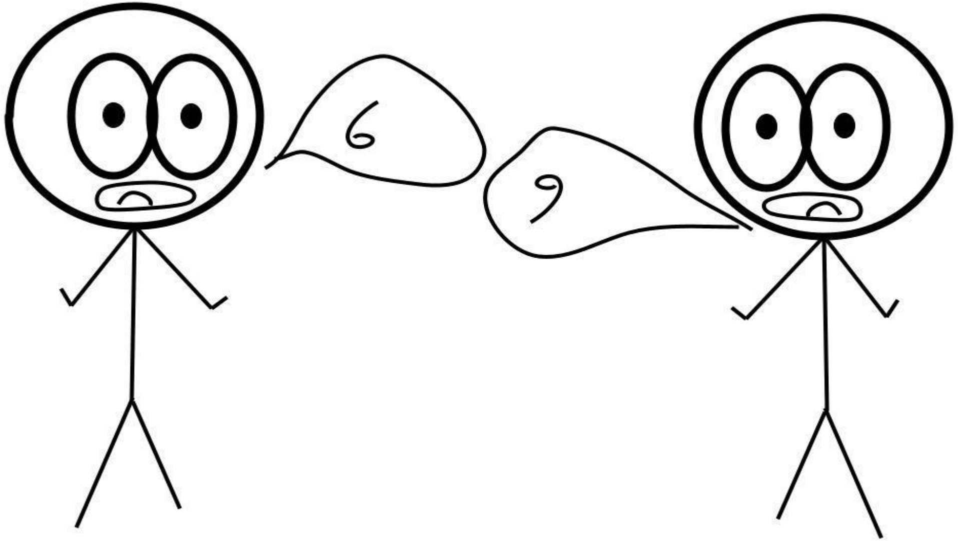
Shifalika Kanwar
Software Engineer
Co-Chair, Distributed Systems Guild

TechAtBloomberg.com

Agenda

1. Why Inclusive & Collaborative Code Review is important?
2. How to promote inclusive code reviews
3. Best practices

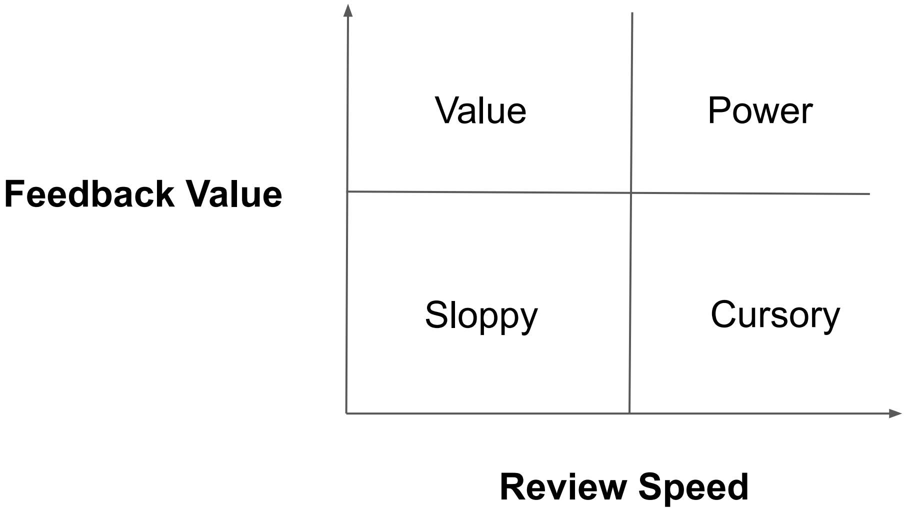
Effective Code Reviews: Socio-Technical Engineering Practice



Reading modes

- Ad-hoc
- Checklist-based
- Scenario-based
- Test-driven

Review Speed vs. Feedback Value



Value

Power

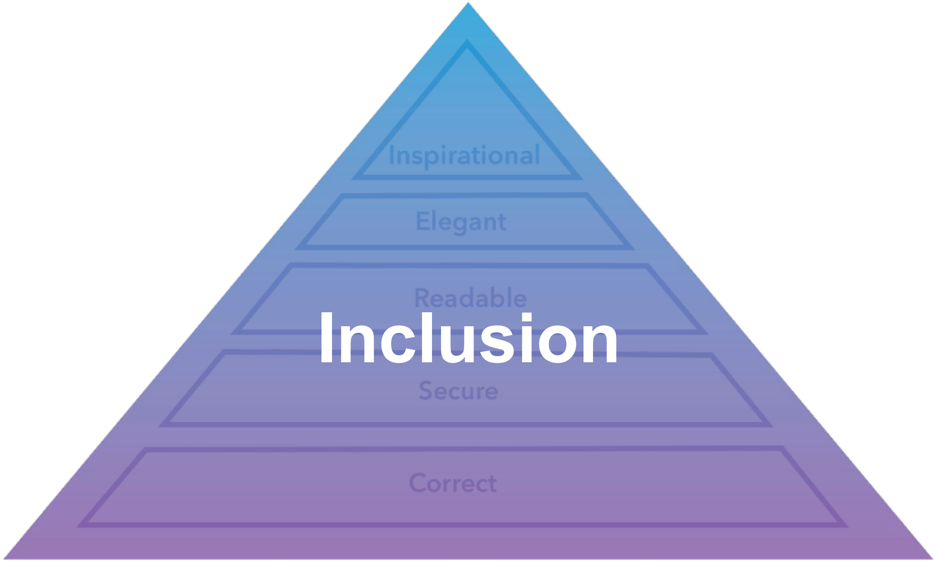
Sloppy

Cursory

Feedback Value

Review Speed

Maslow’s Hierarchy of Code Reviews



TechAtBloomberg.com

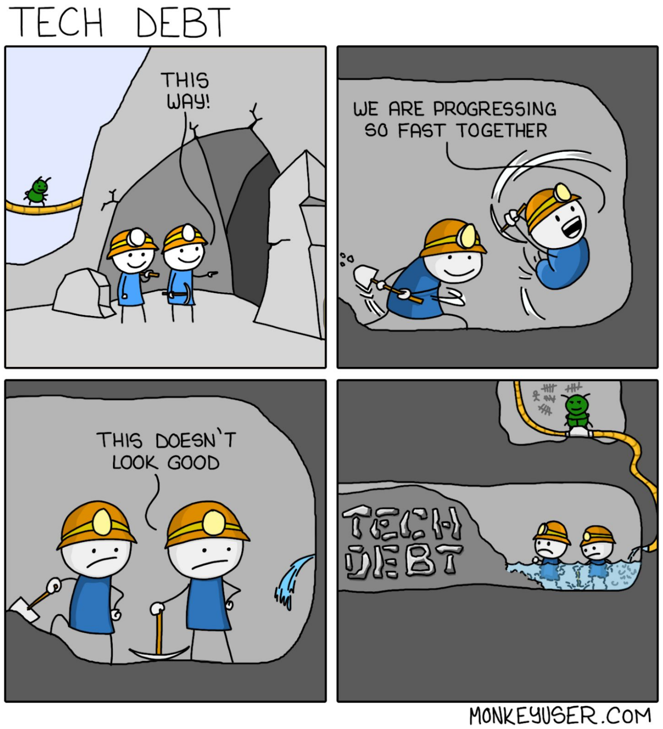
Bloomberg

Engineering

Code review helps to...

- Catch issues early on
- Maintain a consistent codebase
- Spread knowledge
- Create a sounding board / historical record on which to base design discussions ahead of future enhancements
- Create collective ownership

Reducing Tech Debt



https://www.monkeyuser.com/2018/tech-debt/

Frustrations

- Insufficient background info
- Takes too much time and effort
- Unproductive comments
- Feedback drags on
- Valuable feedback might get lost if not documented properly

How do we make the process better?

How do you perceive the code reviewer?

OR

Author Checklist

- Assume positive intent
- Start with a growth mindset
- Conduct your own code review before submitting the PR
- Request code review from two or more people
- Create a well-rounded PR with a template (e.g., problem description, your solution, release mechanisms, test cases, dependencies, etc.)
- Automate formatting and code compliance checks

Author Checklist cont.

- Small PRs are more beautiful
- Plan for big code reviews
  - Set up time to discuss background; Create design documentation; Add review time to project timeline.
- Request a timeline or deadline from 2 or more reviewers
- In case one reviewer is busy, find another
- Respond to vague or unclear comments with collaborative questions

Author Checklist cont.

- Don’t take comments personally
- Ack every comment
- Add unit tests

Reviewer Checklist

- Code review is not a favor
- Decide on an SLA (time frame and PR template)
- Understand the logic or design while reviewing
- Ask clarifying questions
  - Instead of “We don’t need this line” ➔ “Why do we need this line?”
- Use inclusive language
  - Instead of “Your spacing is inconsistent” ➔ “We could keep the formatting consistent by...”
- Do not use vague language
  - Instead of “This code is messy” ➔ “We could make this code easier to read by…”

Reviewer Checklist cont.

- Give actionable feedback with your reasoning
  - e.g., “Break this into two (2) objects, so the interface and implementation are cleaner”
- Review the code, not the person
  - Instead of “You added inconsistent naming” ➔ “To make the naming consistent...”
- For pre-existing issues, find a solution with the author
  - Talk or meet, and record the conclusions
- Remember to give positive feedback
- Break big changes into chunks; book your own time over days for code review
- Use collaborative and empathic language

Reviewer Checklist cont.

- Develop a relationship with the person outside of the tool
- Find a balance between being direct vs. nit-picky or overly critical
- Keep iterating
- Keep an eye on scope creep

Thank you!

We are hiring: bloomberg.com/engineering

Questions?

TechAtBloomberg.com

Bloomberg Engineering