VEROSINT

HUSTEF

HUNGARIAN SOFTWARE TESTING FORUM

The Testers' Secret Weapon - Code Reviews

Bertold Kolics

HUSTEF 2023 - Budapest, October 4, 2023

My Context

- Question Asker, Bulldog Engineer
- **Not a Gatekeeper**
- Verosint - small startup < 20 employees
  - SaaS business
  - We detect & prevent online account fraud
- Past
  - mabl (e2e test automation as a service) - small
  - CS Disco (legal tech) - medium
  - ESO Solutions (healthcare) - medium
  - Ping Identity / UnboundID - small/medium
  - Sun Microsystems - large
  - SZTAKI - medium

Definition

- **What**: review code for completeness and correctness
- **When**: prior to integration into product
- Form of *static testing**
- Informal vs. formal
  - mind the context
- Synchronous
  - in-person/virtual walkthrough
  - peer or mob programming
- Asynchronous
  - out of band review typically in the form of a pull request
- Participants:
  - author: engineering (development, test, even ops)
  - reviewer: engineering, ops, technical writers

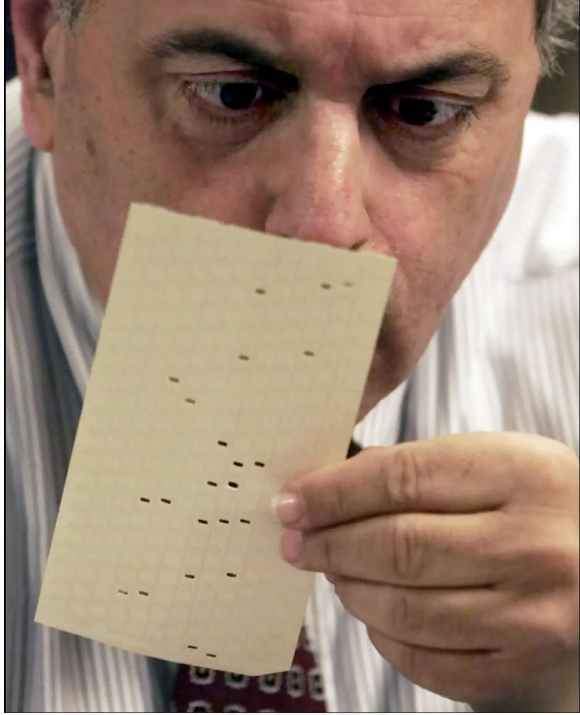


Image credit: Wilfredo Lee/AP/REX/Shutterstock.com

HUSTEF
HUNGARIAN SOFTWARE TESTING FORUM

Why

Accelerate the Achievement of Shippable Quality

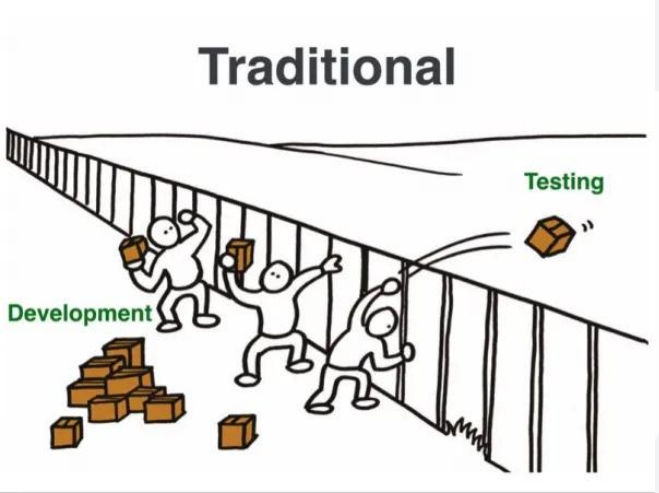
source: https://www.moderntesting.org/

Code reviews:

- Create shared understanding
- Reduce rework
- Help instill best practices
- Opportunity to learn from others, teach & train
- Influences outcome
- Gels the team when done properly as we rarely work in isolation

Remember: soft skills are the hard skills

Case Study: Sun Microsystems



- Year: **2003**. Event: acquisition of Waveset.
- Role: developer
- QA team and technical writers under separate management structure.
- Product: highly-customizable identity provisioning, on-prem
- Process: **throw it over the wall**
- Initial state
  - frequent patch releases (multiple times a week), regressions
  - implementation with lots of shortcuts, unsuitable for quick ramp of new customers
  - core engineering involvement

Image credit: John Anju

Case Study: Sun Microsystems - Review Process

- End state
  - max once a month patch release managed by sustaining team
  - design reviews for features
  - longest code review template ever
  - significantly higher quality / throughput

- Code review template **forced developers** to
  - **communicate** and create shared understanding
  - **slow down**: have you heard “slow down to speed up” before?
  - carefully **consider** all aspects of the changes
  - get **buy-in**: plan changes first, code next

Activity Is Not Impact

Build the right it & build it right.

HUSTEF
HUNGARIAN SOFTWARE TESTING FORUM

Case Study: Sun Microsystems - Process

- Developer
  - clarifies requirements if necessary with product or engineering manager
  - consults test specialists if necessary
  - implements increment including tests and changes to documentation
  - tests code manually and with automated tests
  - fills out code review template
  - solicits feedback, revises code
  - integrates change
  - addresses any test failures
  - updates ticket in issue management system: details, links to source changes, etc

- Template reflecting the above process

Case Study: Sun Microsystems - Template

- Contents:
  - Summary of the issue
  - Alternatives considered
  - Details about solution implemented
  - Impact (cross-cutting concerns)
    - performance, scalability, security, accessibility
  - Net new tests implemented
  - Release notes and notes on updating product documentation
- Details about implementation copied into issue manager
  - QA team - all over the moon 😍
  - Technical writers - no longer felt neglected

Image credit: Teams of Distinction

Case Study: ESO Solutions

- Year: 2017. Context: leading QA practice for multiple development teams.
- Company: ~200 employees
- Product: electronic health record, SaaS, US customers only
- Team structure:
  - cross-functional, 2-pizza sized: product manager, developers, test engineers
  - matrix organization
  - 2 test engineer / team: SDET and traditional QA
- Process: scrum with planning/standup/retro ceremonies, 2-week cycle
- Key elements:
  - co-located teams - sitting in close proximity
  - review format: most often pairing driven by short delivery cycles
  - evolving test automation engineers to own framework instead of feature automation

In The Trenches

Practices that worked for me

- Don’t lose sight of the goal
- Don’t make it personal
- Be curious
- Teach & learn
- Pick your battles
- Size matters
- Time - yours and theirs, interruptions
- Reviewed code is not bug free code

Remember: more time spent on reading code than writing

Image credit: [png image from pngtree.com]

Extent of Code Changes

- The brain cannot keep focused on large code changes
  - too much changes lead to subpar reviews
- Keep code changes **under 400 lines**

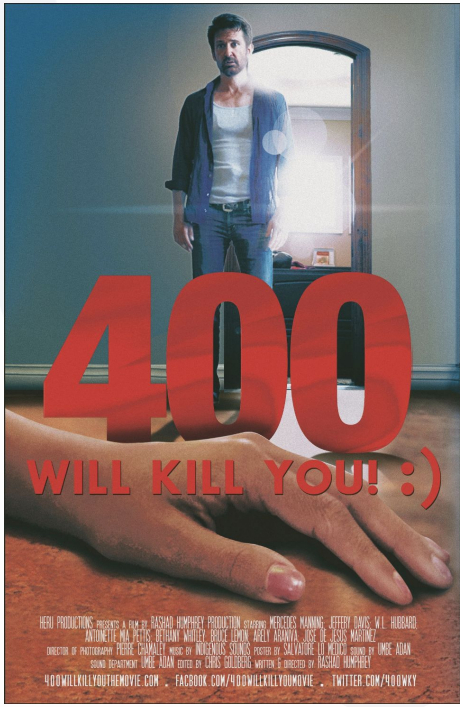


Image credit: IMDB

Code Walkthroughs

- Consider code walkthroughs for new features, large changes
- Best if accompanied by
  - READMEs
  - Drawings / sketches
  - Example code / automated tests
  - Whiteboard
- In person or over Zoom
- Allow time for questions

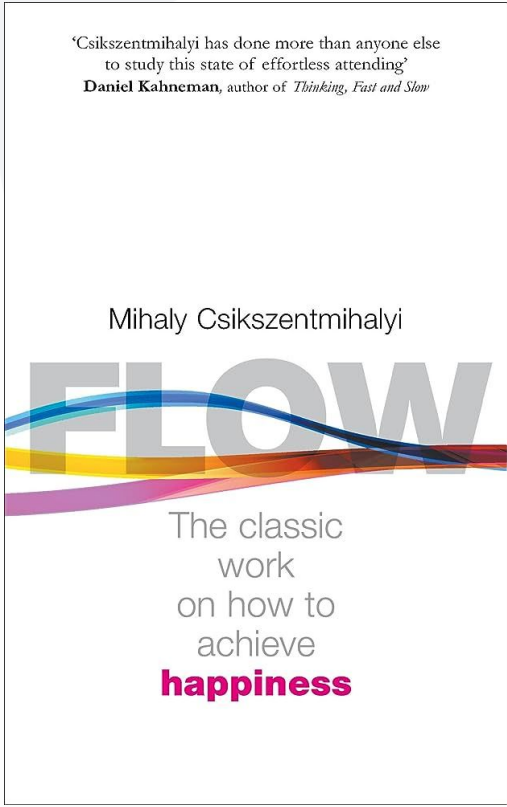
Before Code Reviews

- Run static analyzers
  - Helps reviewers focus on **higher level concerns**
- Ensure that automated tests are **passing**
  - Helps reduce the requests for reviewing code changes again

Image credit: Wikipedia

Avoid Interruptions

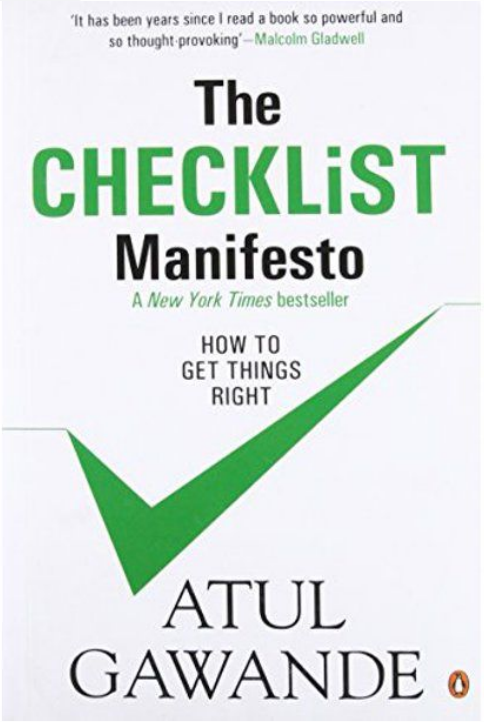
- Interruptions are productivity killers
- Bundling technique
  - work interruptions limited to certain days or part of the day
  - for example: code reviews daily after standup
- Get your ducks in the row before requesting review
  - respect the time and **interruptions** code review
- Anti-pattern example:
  - requesting a review multiple times for multiple revisions in the same pull request



HUSTEF
HUNGARIAN SOFTWARE TESTING FORUM

Checklists / Templates to The Rescue

- Sets expectations
- Easier for reviewers to follow a consistent format
- May be specific to a repository

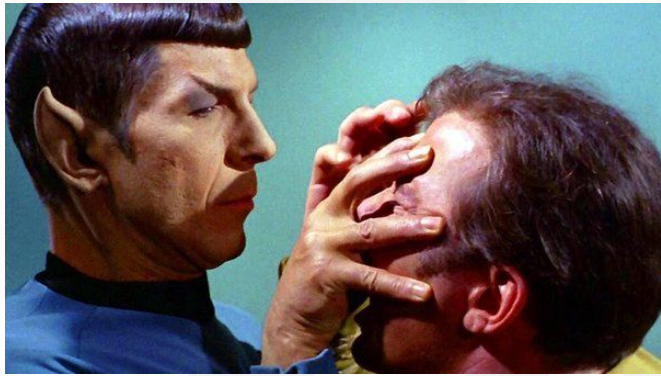


HUSTEF
HUNGARIAN SOFTWARE TESTING FORUM

Exceptions Are OK, too

- Consider relaxing code review requirements without sacrificing quality
- Examples:
  - 1-line code change
  - Changing only message catalog keys
  - Updating project README

Overcommunicate



- Don’t forget to set the context
- Links are helpful
  - tickets
  - design documents
  - decisions records
- Add your own code review comments in areas
  - where implementation choice may not be clear
  - where additional considerations were needed
- Eliminates back-and-forth
- Sometimes I mention: “I am not a Vulcan, I cannot mind-meld”
- Anti-pattern example:
  - 10-line code comment to explain a complex conditional

What’s In It For Me?

Testing Specialists

- Make yourself more valuable to the team
  - coaching developers on testing
  - removing yourself eventually as the bottleneck
  - shorten the feedback loop
- Learn programming
  - reading is easier first than writing
- When done properly
  - code reviews create another collaborative venue
  - foster shared understanding
  - gel the team

Developers

- Opportunity to learn testing techniques
  - testing - especially feature-level - is a **must-have skill**, not a dedicated job
- Opportunity to share knowledge between developers
  - junior developers, interns can learn best practices to use, patterns to follow
  - senior developers can exercise their coaching, technical leadership skills
- Less rework, more new work
  - do you prefer fixing bugs or develop new features? Most will pick the latter.
- Think value
  - you are not valued on the # of lines of code you produce
  - reviews help to focus our efforts

Leaders

- Make it part of the software development lifecycle
  - will also pay dividends when customers ask about development practices
- Code reviews can
  - gel the team,
  - provide growth professionally to team members,
  - increase the value the team delivers

Wrapping It Up

Key Takeaways

- Code reviews are an essential part of the software development life cycle
- Can be done in many shapes and forms
  - having an agreed upon process appropriate for the context is key
- When done properly, it
  - helps create shared understanding,
  - becomes a powerful teaching tool,
  - provides a platform for coaching testing as well as development skills,
  - gels the team

Resources

- My Code Review Guidelines
- Newsletters
  - The Pragmatic Engineer by Gergely Orosz
  - The Beautiful Mess by John Cutler
- Podcasts
  - A/B Testing
  - Testing Peers
  - Maintainable Software
- Books
  - Your Code As A Crime Scene (2nd edition coming)
  - Software Design X-Rays
  - Clean Code: A Handbook of Agile Software Craftsmanship
  - SmartBear's Best Practices and e-book
  - User Story Mapping
  - Accelerate

bit.ly/hustef2023

HUSTEF
HUNGARIAN SOFTWARE TESTING FORUM

Thank you