Code Review 3base

Ran Tavory, Appsflyer
Oct 2020

whoami

Motivation First

- For the Customers
- For Us

For the customers

The immune system

*"Running **Continuous Deployment** without an **Immune System** is playing **Russian Roulette**"*

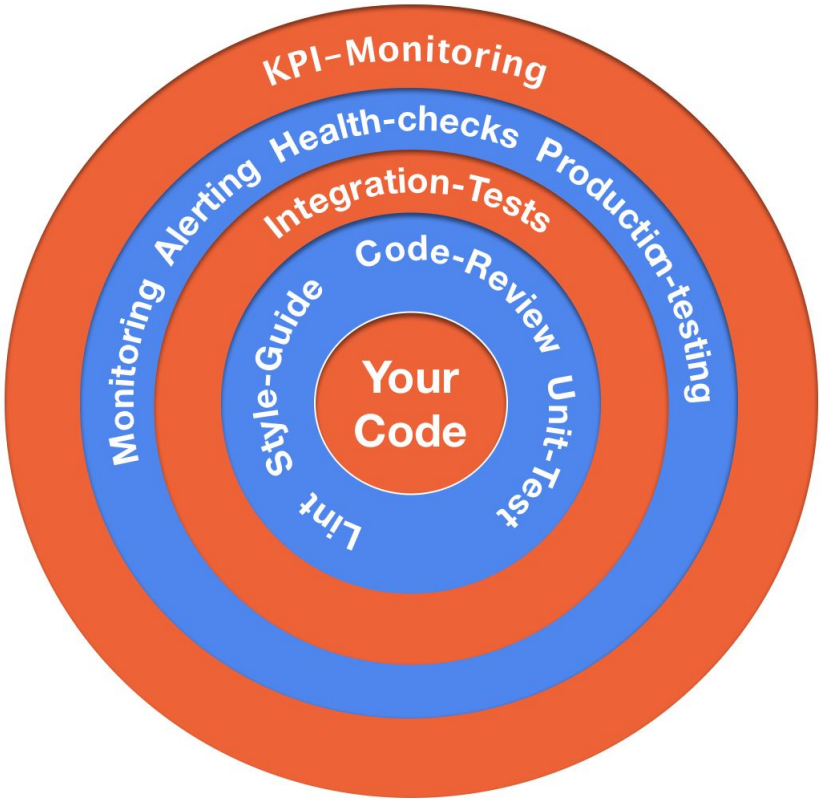
The Immune System

Any line of defence will eventually get compromised

Reminder: the Bar-Lev line of defence

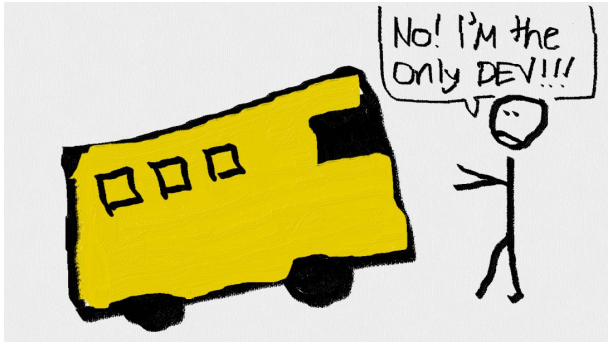
=> Solution: Multiple lines of defence

The SAAS Immune System



Code review - For us

- Improve our skill
  - Great way to **Teach**
  - Great way to **Learn**
- Maintain **Coding conventions**
- **Find Defects** early
- More devs familiar with the code
  - At least two principle
  - Increase bus factor



Programs must be written for people to read

- Harold Abelson

But if we don't **test** code readability, how do we know it is readable?

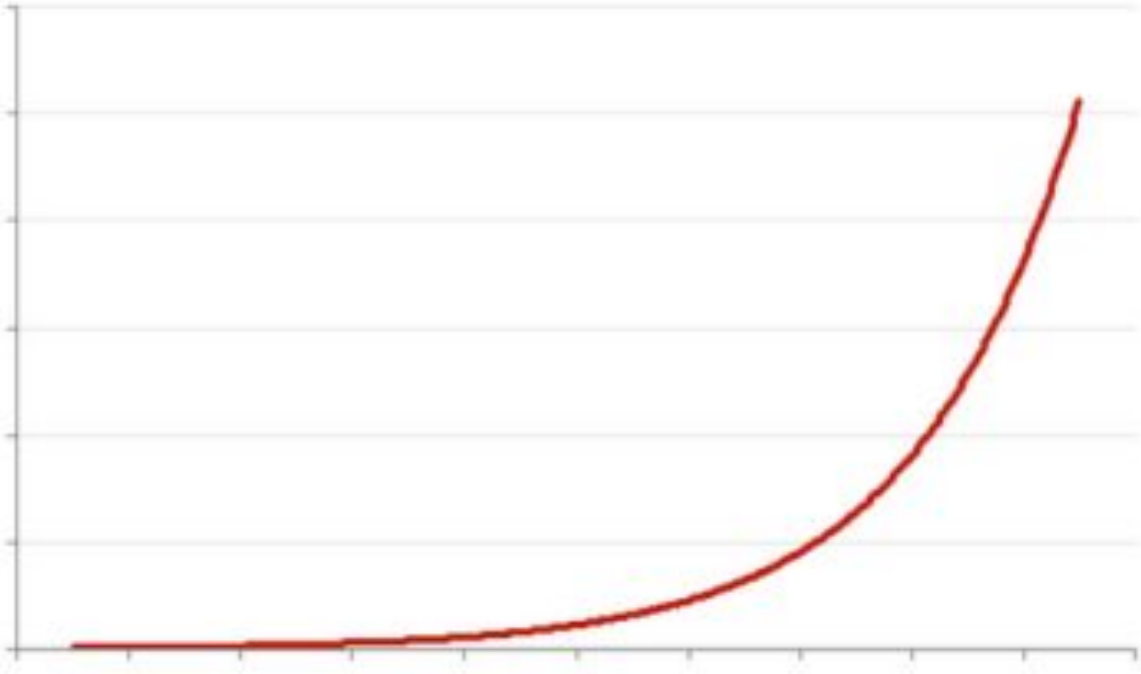
Quality v/s Speed ?

It depends

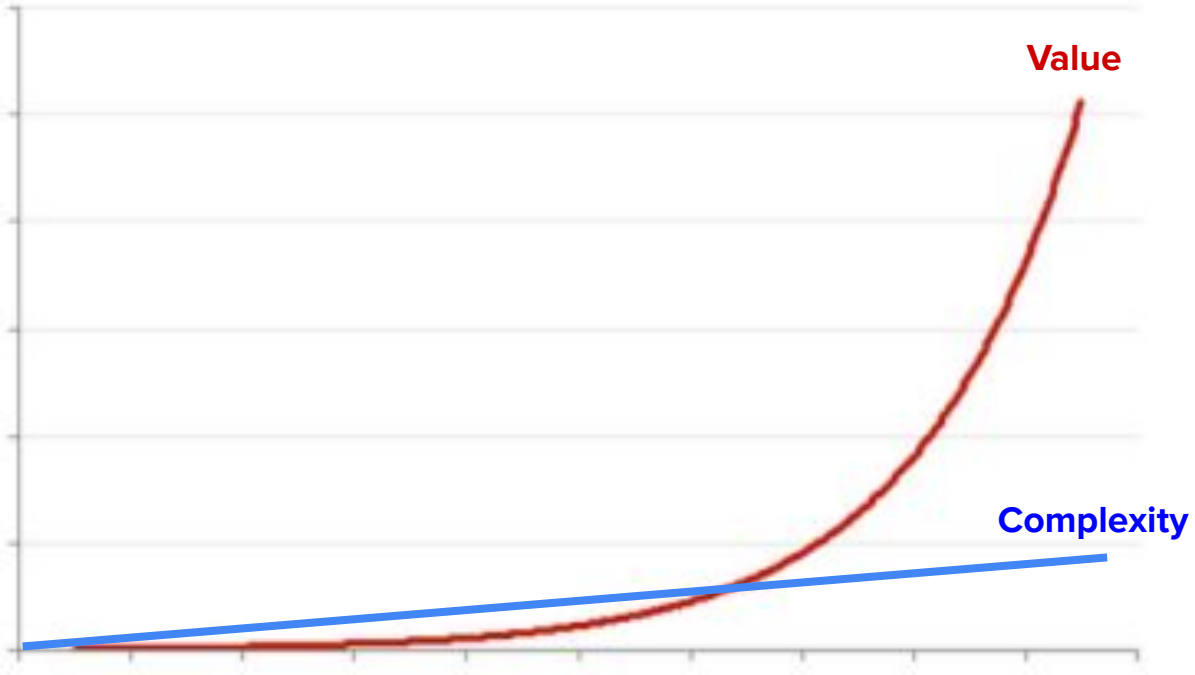
Quality is a requirement in order to maintain speed

WHY?

Exponential Value (and complexity)



Exponential Value but linear complexity



The Art of Writing is the Rewriting

Similar to nobles and poetry

80% of code will be rewritten

Conclusion: We have to **optimize for readability**

Fun Quotes

Programs must be written for people to read, and only incidentally for machines to execute.

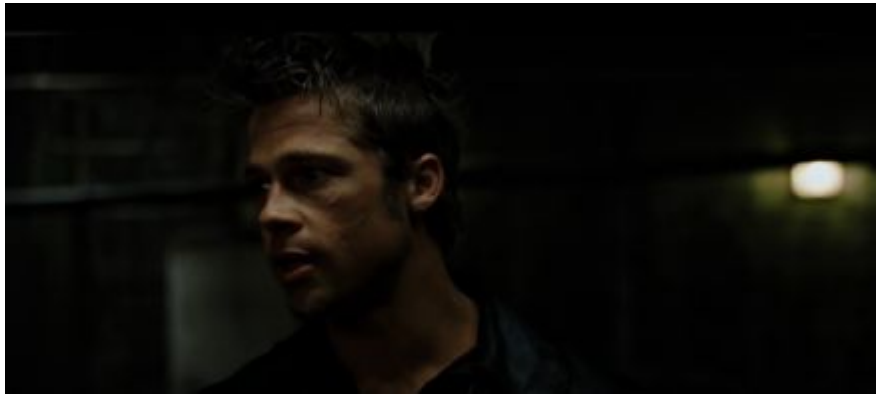
–Hal Abelson and Gerald Sussman

Simplicity is prerequisite for reliability

—Edsger W. Dijkstra

Best Practices

The first rule of code review



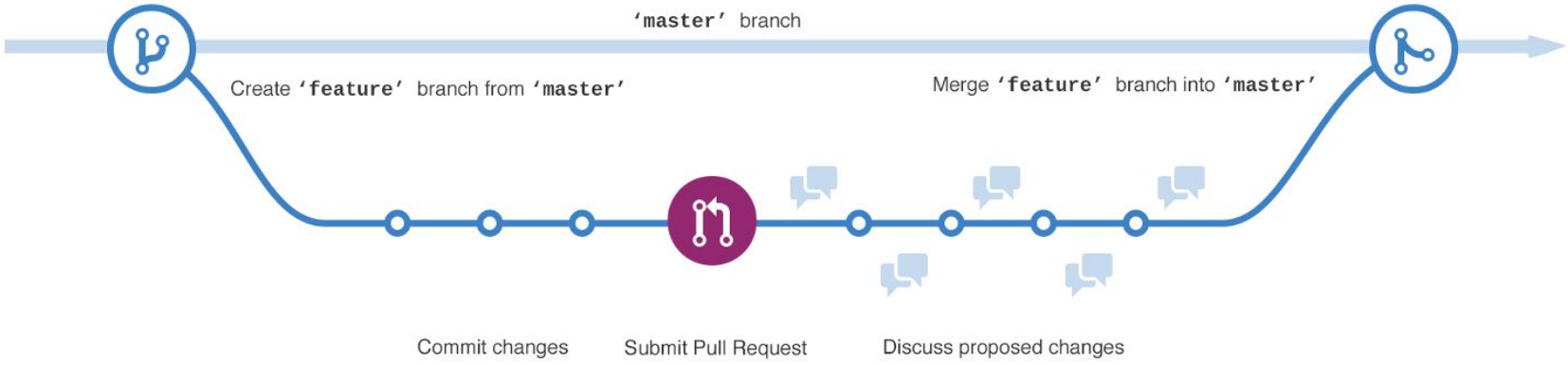
The first rule of code reviews is:

- **There are no rules in code reviews.**

Code Review Objectives

- Correctness
- Maintainability
- Shared knowledge
- Sharpen skills

The Flow


'master' branch
Create 'feature' branch from 'master'
Merge 'feature' branch into 'master'
Commit changes
Submit Pull Request
Discuss proposed changes

Committer

The committer's flow
High level

1. Create a branch
2. Push
3. Create PR / MR
4. Assign to reviewer
5. ... back and forth ...
6. Approval
7. Merge

Before review

Committer duties

- Pull master
- Branch from master
- Small and atomic change *
- Lint
- Test
- Review **
- Rebase
- Push
- Open PR / MR***

Review your commits

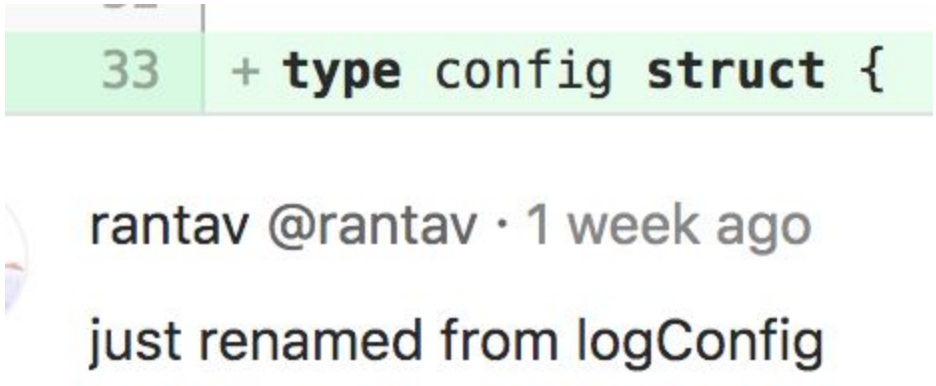
Committer - before push

- **Review** your commits and commit messages.

Open a Merge Request

Committer duties

- **Review** the diff. (again)
- **Annotate** (add comments)
- Write a descriptive **message**
- Assign to reviewer

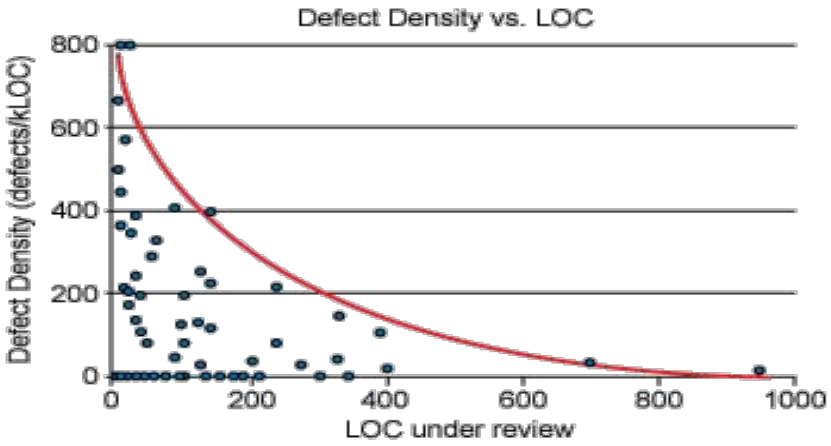


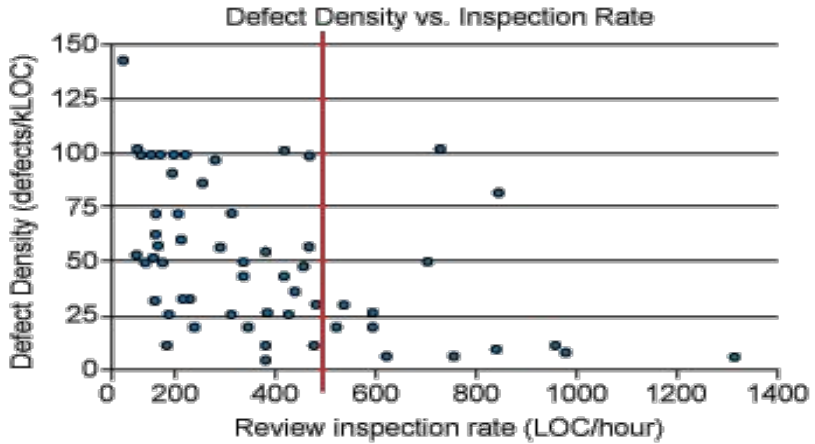
Annotate your MR

Small is Good

Strive for small PR / MR

How Small “Small”?





How small?

- < ~400 LoC
- **< 1h** of reading
- **Atomic** changes

Reviewer

The reviewer flow
High level

1. Be informed
2. Review Code
3. Leave Comments
4. ... back and forth
5. Approve when Happy

Before Review

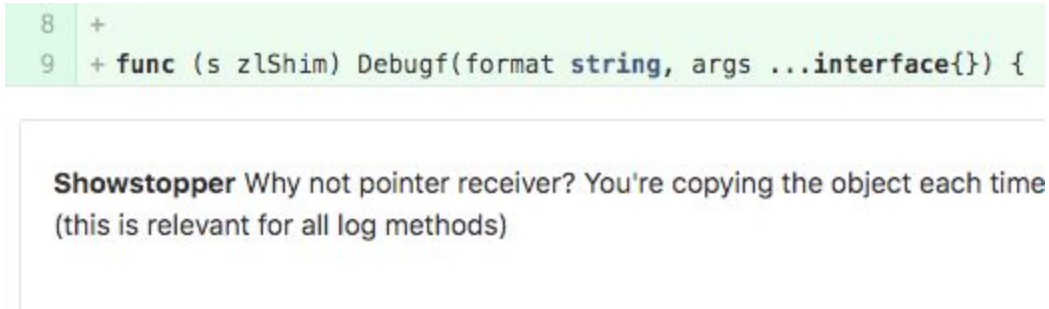
Reviewer duties

- Get to know the **technology** (database, UI framework...)
- Get to know the **product** (and use case)

During Review

Reviewer duties

- High priority
- Read code
- (fetch and run locally)
- Leave comments * (mark showstoppers)
- Add a summary



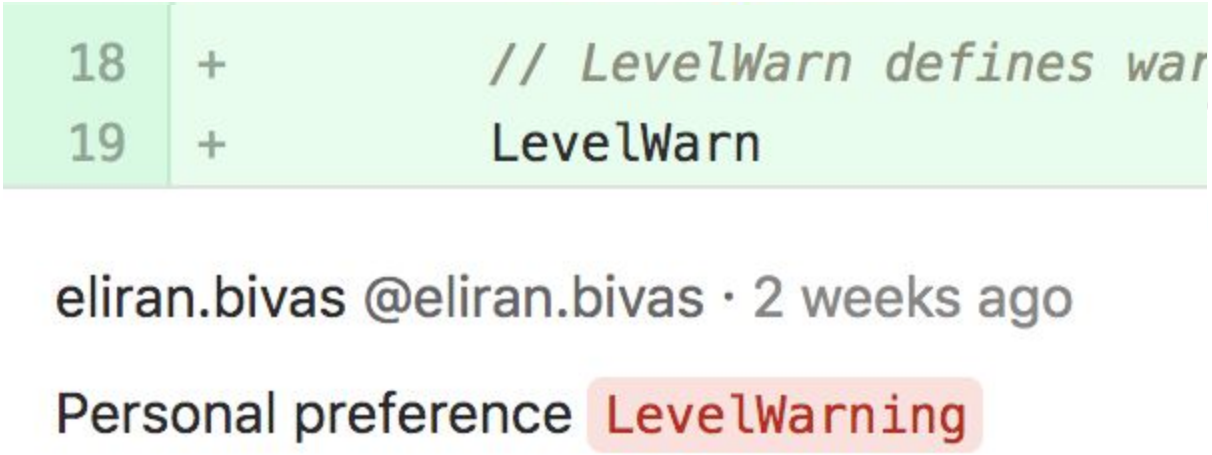
Mark showstoppers

Adding Comments

Reviewer

- Respectful and punctual.
- Suggestive, not decisive
- Add references
- Be explicit.
- Personal preferences / style?

What's wrong with this?



Personal preference / style

What's wrong with this?

Be suggestive

```
9  + func (s zlShim) Debugf(format string, args ...interface{}) {
```

Why not pointer receiver? You're copying the object each time this method is called (this is relevant for all log methods)
https://github.com/golang/go/wiki/CodeReviewComments#receiver-type

Add references

The Checklist

- **Think**
- Correctness & Completeness
- Naming and Style
- Test
- Manual tests / MSP namespaces link
- Deleted tests
- Commented out code
- Encapsulation, modularity, coherence
- DRY
- SOLID principles
- Edge cases
- Security
- Race conditions
- Documentation
- Monitoring
- Logging
- Configuration
- DB queries
- Performance
- ...

Provide Positive Feedback

Reviewer duties

- Awesome work
- Niccccceeeeee!
- Love the implementation

During Review

Committer duties

- Address & **resolve** comments (discussions)
- Be **accepting**. "*Good call. I'll make that change*."
- Don't take it personally. **The review is of the code, not you**.
- **Learning** opportunity
- **Teaching** opportunity
- "I didn't understand. Can you clarify?"
- Fix, rebase, push & notify

Timeline

Don’t drag

Key point:

- Lengthy reviews processes are disastrous
- Reviewer: Same day / Very high priority

Offline Reviews

When to go offline? (f2f)

- Start online on GL
- If too many back-and-forth, go offline (f2f)
- At the end document the offline conversation on GL so that next person gets your PoV

Guiding Principles

Maintainability » Speed

Reading ≫ Writing

Clear » Clever

Simplicity

Fun Quotes (again)

If something unusual is happening, leave evidence for the reader

–Brian Kernighan

Simplicity is prerequisite for reliability

—Edsger W. Dijkstra

Styleguides

- Clojure: https://github.com/bbatsov/clojure-style-guide
- Javascript: https://github.com/airbnb/javascript
- Go: https://github.com/golang/go/wiki/CodeReviewComments

References

- https://smartbear.com/SmartBear/media/pdfs/best-kept-secrets-of-peer-code-review.pdf
- https://dave.cheney.net/paste/clear-is-better-than-clever.pdf