



# Benefits of Automated Code Review

September 25, 2019

Matt Kohler

[matt@codacy.com](mailto:matt@codacy.com)





# 01 What is code review?

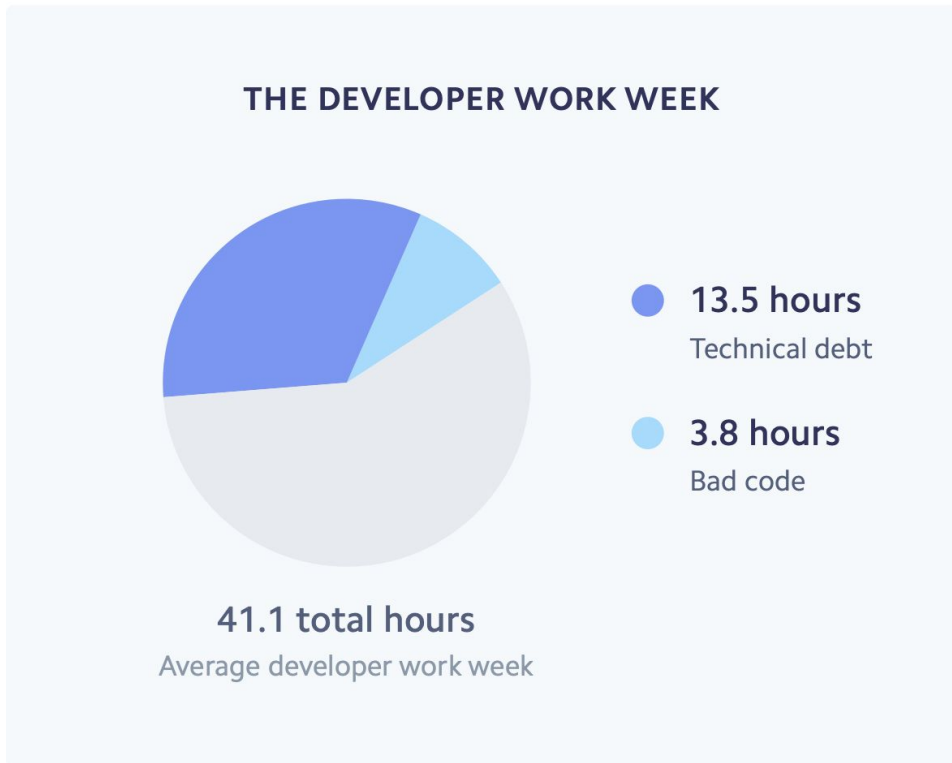


# Code Review

- A process that reviews code using a predefined set of rules /standards
- Primary goals
  - Identify errors
  - Follow best practices
  - Maintain code standards
  - Capture security issues



# How is your time spent?



**~\$85 Billion**

**Global GDP loss from  
developer time spent  
on bad code annually**

Sources: Evans Data Corp., CIA Factbook, Stripe research



## 02 Automated vs Manual



## Manual Code Review

- Manual code review is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Often done by a peer or a QA team
  - Developers submit their code for review and issues are sent back
- Generally run through a checklist to identify issues
- Feedback can be personal
- Conversations should focus on how code is written
  - Code logic
  - Can this be written more effectively?
  - Understandable



# Automated Code Review

- A process that reviews your code **without** manual input using a predefined set of rules
- Automated vs manual code review
  - Time spent reviewing
    - limited capacity of a human vs code volume
  - Time spent waiting
    - team workflows and time zones
  - Human error
    - personal biases and interpersonal communications
  - Quality Assurance
    - Human vs machine capacity



# 01 What is Static Code Analysis?





# Static Code Analysis

- A way to review your code that **does not** execute any code
- Linters are technically SCA tools, but **do not offer**:
  - Technical debt avoidance / planning
  - Security screening
  - Duplication avoidance
  - Complexity metrics
  - Coverage integration
  - Compliancy
  - Education on best practices
  - Automatic maintenance of rules

# Open source SCA tools

Python	yala	formerly	Elixir	credo	Go	golangci-lint	gometalinter	Haxe	haxe-checkstyle	Dart	linter
	pylint	flake8		jinjalint		golint	go vet		Rust		rust-clippy
Scala	Lintor	scalastyle	HTML	bootlint	Puppet	clang-format	puppet-lint	Graph	graphsql	Dockerfile	dockerfile_lint
	scapegoat	wartRemover		htmlhint		clang-tidy	polylint		Lua		lua-lint
Java	findbugs	checkstyle	Markdown	markdownlint	Crystal	cppcheck	ameba	Sass	scss-lint	styelint	postcss-bem-linter
	pmd	uncrustify		remark-lint		oclint	polylint		Sass		sass-lint
JavaScript	eslint	clinton	Pug	mdl	C/C++	uncrustify	Crystal	Sass	alex	English	proselint
	prettier	jshint		pug-lint		uncrustify			Mess Detector		proselint
	xo	standard	Ruby	rubocop	Objective-C	oclint	PHP	phplint	Perl	perlcritic	CSS



## Tools in the code review ecosystem

- Continuous Integration tools (like Jenkins or CircleCI)
  - With OS linters
  - Or coverage reports
- Pure code coverage tools (Clover, CodeCov, Coveralls)
- Project management tools that aid manual code review



## Using multiple tools leads to inefficiency

- Team members spend time configuring tools rather than coding
- Costs are multiplied across multiple tools
- Tools may not all integrate with each other
- Constant maintenance of standards required
- Context switching for developers reduces efficiency



## 03 Use a code review suite



## Organizational Benefits

- **Efficiency:** Tools like Codacy shows managers what to prioritize. Less time reviewing means more time coding.
- **Code quality standardization:** These tools provide preset, customizable standards that can be enforced.
- **Code quality predictability:** Duplication and complexity metrics can tell you where problems might develop before they actually do.
- **Team management:** A suite provides easily-accessible metrics for non-technical and management-level reports.
- **Proactive:** PRs that will have a detrimental impact on code quality can be blocked



## 04 Codacy Demo