Carleton University

Reimagining Code Review in the Age of AI

Olga Baysal

olga.baysal@carleton.ca

olgabaysal.com

@olgabaysal

Consortium for Software Engineering Research
May 29-30, 2025

the MENU

APPETIZERS

Why Code Review Still Matters
A Brief Evolution of Code Review

ENTREE

A Theory of Code Review
Why is code review hard?
Why is it error-prone?

DESSERT

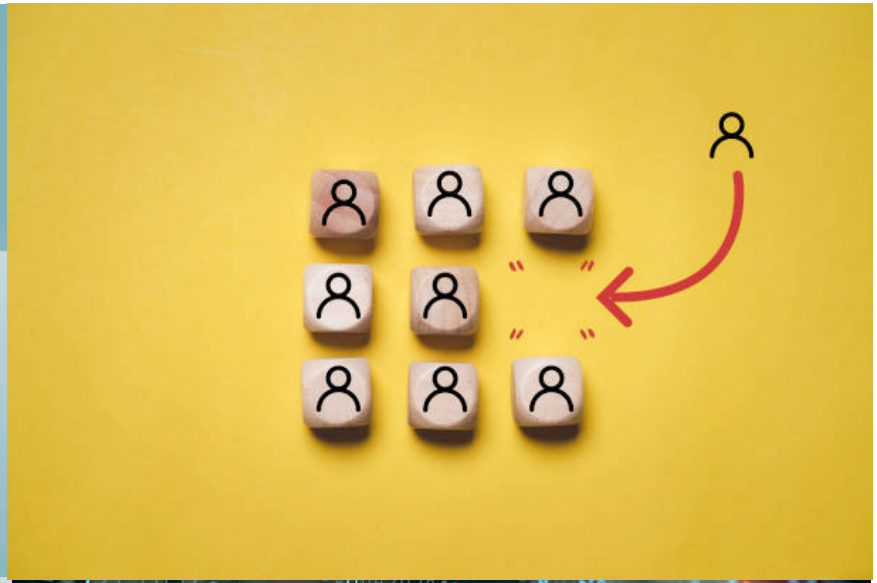
The New Frontier - AI in Code Review

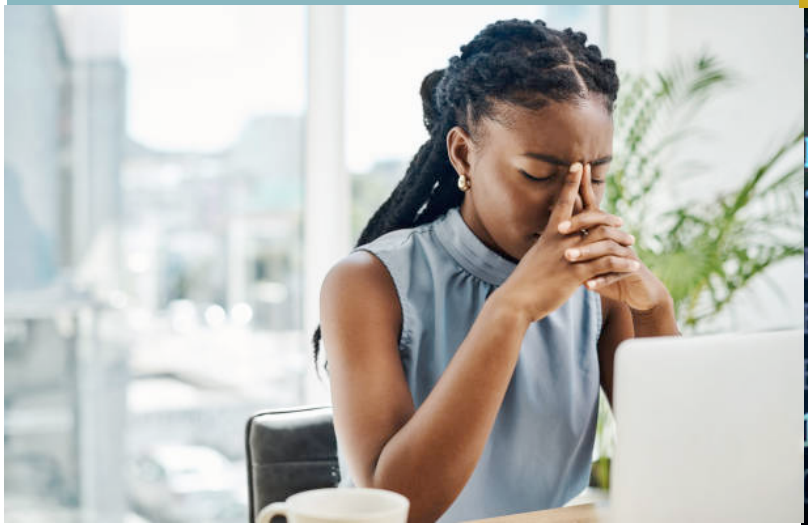
Code Review

Assessing the quality of submitted code changes

Why Code Review Matters

Challenges






Data Breach
Cyber Attack
Protection Failed
System Safety Compromised

Evolution of Code Review

- **1970s–1980s: Formal Manual Reviews**
  - Fagan Inspections
  - Paper-based, in-person, structured roles, heavy documentation

- **1990s–2000s: Email & Patch-Based Reviews**
  - OSS mailing list patches (e.g., Linux kernel)
  - Asynchronous, informal

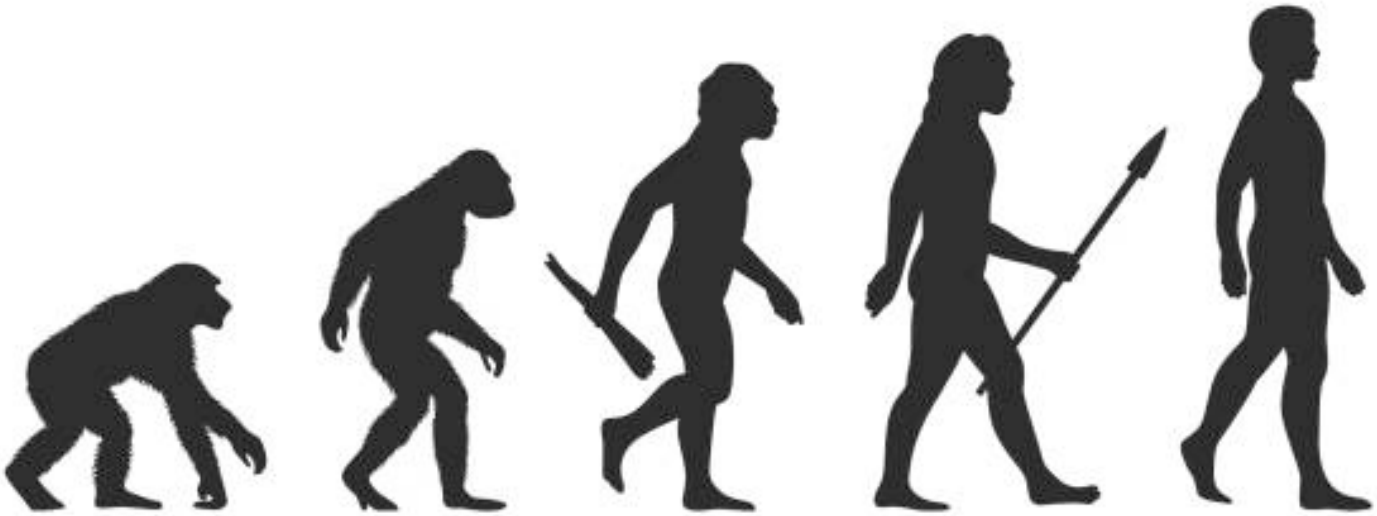
- **2000s–2010s: Web-Based Code Review Tools**
  - GitHub PRs, Gerrit
  - Inline comments, diffs, CI hooks, integrated into version control

Evolution of Code Review

- **2010s–2020s: Collaborative (Modern) Reviews**
  - Frequent, fast, and lightweight reviews
  - Integration with CI/CD pipelines

- **2020s–Now: AI-Powered Code Reviews**
  - LLM-based suggestions (e.g., GitHub Copilot)
  - Auto-detection of bugs, style, and logic issues

Evolution of Code Review



Formal
Centralized
Manual inspections
Bug-finding

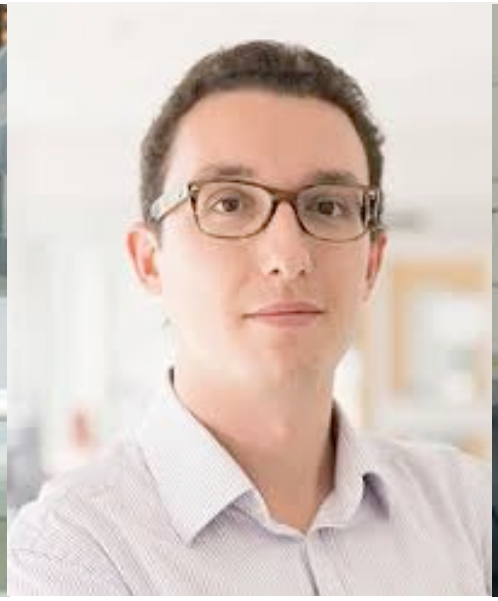
Augmented
Distributed teams
Automated assistance
Collaboration/learning

Does Code Review Matter Today?

- CHASE 2025: 2 papers
- ICPC 2025: 2 papers
- Doctoral Symposium: 2 papers
- MSR 2025: 3 papers
- ICSE 2025: 5 papers

Keynotes' Takeaway

CHASE 2025



MSR 2025

ICPC 2025

Need for more **theories** in SE!

Theories of Programming

- Amy Ko (University of Washington - Seattle, US)
- Thomas D. LaToza (George Mason University - Fairfax, US)
- David C. Shepherd (Virginia Commonwealth University - Richmond, US)
- Dag Sjøberg (University of Oslo, NO)

https://medium.com/bits-and-behavior/dagstuhl-trip-report-theories-of-programming-382543a3e540

Developers make mistakes during code review

Code Review Quality

Gandalf, the reviewer

Study I: Code Review Practice

Investigating Code Review Quality:
Do People and Participation Matter?

Olexii Kononenko*, Olga Baysal†, Latifa Guerrouj‡, Yaxin Cao†, and Michael W. Godfrey*
*David R. Cheriton School of Computer Science, University of Waterloo, Canada
Email: okononen@uwaterloo.ca, migod@uwaterloo.ca
† Department of Computer Science and Operations Research, Université de Montréal, Canada
Email: yaxin.cao@umontreal.ca, olga.baysal@umontreal.ca
‡ Département de Génie Logiciel et des Technologies de l'Information, École de technologie supérieure, Canada
Email: latifa.guerrouj@etsmtl.ca

Preprint

Kononenko et a., ICSME 2015

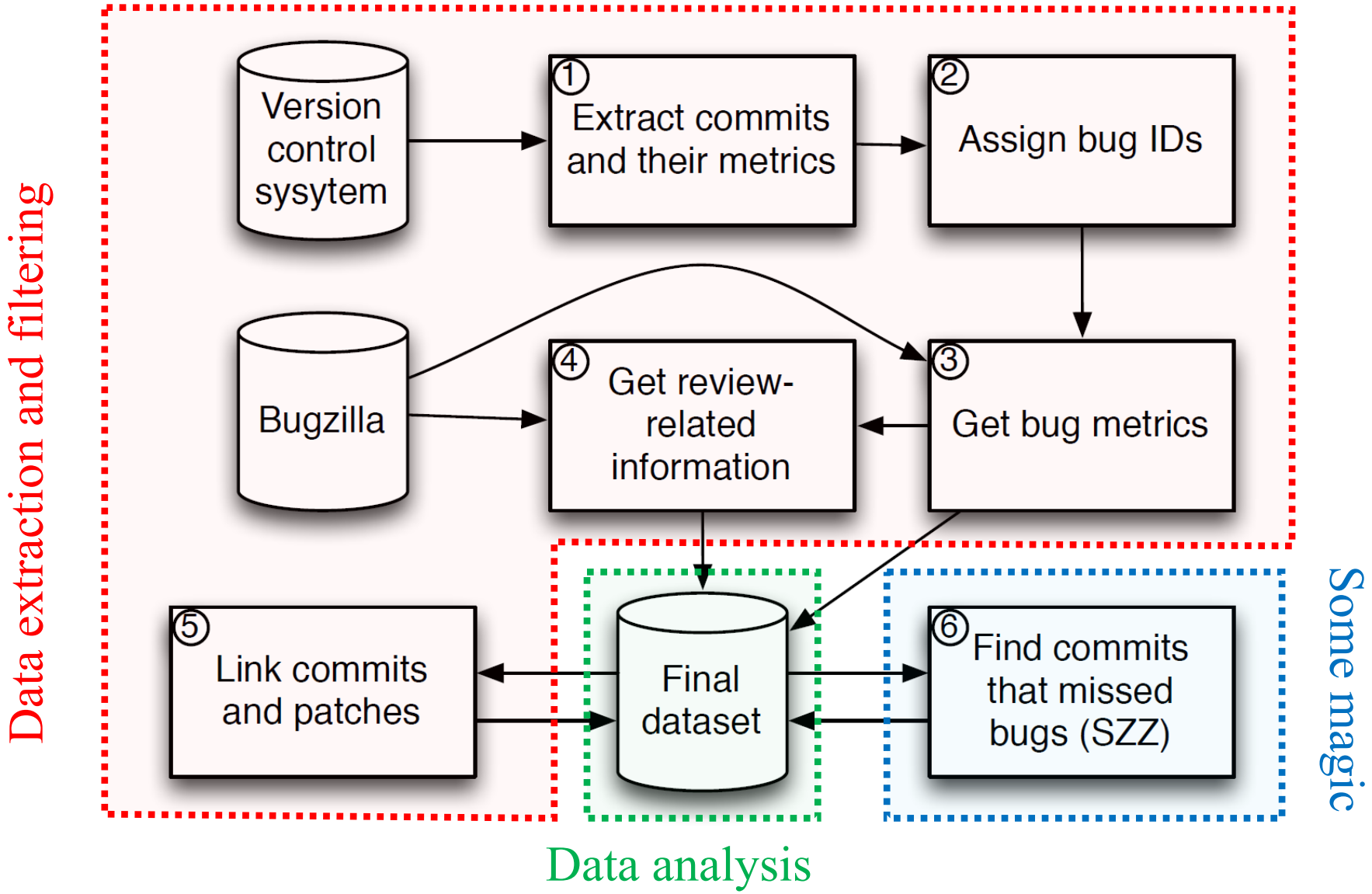
- Mining development repositories: issue tracking, version control system, code review
- **Do developers miss many bugs?**
- **What factors affect code review quality?**

Case Study Setup

- Mozilla project
  - Commits to the *mozilla-central* repository (from 2013-01-01 to 2014-01-01)
  - Bug report data from Bugzilla
  - Code review information from Bugzilla

System	Commits	Reviews	Writers	Reviewers
Mozilla-all	27,270	28,127	784	469
Core	18,203	18,759	544	362
Firefox	2,601	2,668	214	110
Firefox for Android	2,106	2,160	108	72

Data Mining


Data extraction and filtering
Version control sysytem
1 Extract commits and their metrics
2 Assign bug IDs
Bugzilla
4 Get review-related information
3 Get bug metrics
5 Link commits and patches
Final dataset
6 Find commits that missed bugs (SZZ)
Some magic
Data analysis

SZZ Overview

- The SZZ algorithm provides a list of commits that led to a bug (as indicated by a given bug-fix commit)

- "Led to a bug" example:

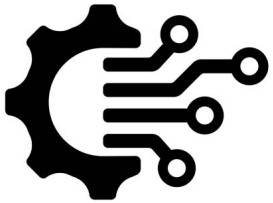
Before (bug-inducing)

```
if (foo == null)
  bar(foo);
```

After (bug-fix)

```
if (foo != null)
  bar(foo);
```

Factors and Metrics



Technical	Personal	Participation
Size (LOC)	Review queue	# of developers on CC
# of chunks	Reviewer experience	# of comments
# of files	Writer experience	# of commenting devs
Module	Reviewer experience for module	Avg # of comments per developer
Bug priority	Writer experience for module	# of reviewer's comments
Bug severity		# of writer's comments
Super review required		
# of previous patches		
# of writer's previous patches		

Do Reviewers Miss Many Bugs?

System	Number of reviews	Number of buggy reviews	% of buggy reviews
Mozilla-all	28,127	15,188	54.0%
Core	18,759	10,184	54.3%
Firefox	2,668	1,447	54.2%
Firefox for Android	2,160	1,210	56.0%

Previously reported findings on the % of buggy commits:

- Kim et al. – from 10% to 74% depending on a project;
  30% for Mozilla (2003-2004 commit history)
- Sliwerski et al. – 42% for Mozilla (commits prior 2005)

Do Personal Factors Affect Code Review Quality?

	Mozilla	Core	Firefox	Firefox for Android
Adjusted R^2	0.128	0.123	0.173	0.138
Size (LOC)	0.102***	0.098***	0.108***	0.115***
Chunks	†	†	†	†
Number of files	0.058***	0.059***	0.109***	0.062*
Module	⋆	n/a	n/a	n/a
Priority	⋆	⋆	‡	·
Severity	‡	‡	·	‡
Super review	-0.139**	-0.177***	·	n/a
Review queue	0.017***	0.0204***	0.038**	0.045**
Reviewer exp.	-0.013***	-0.012***	-0.029***	-0.041***
Reviewer exp. (module)	†	†	‡	0.018*
Writer exp.	·	-0.004*	‡	‡
Writer exp. (module)	†	†	‡	·
# of previous patches	†	†	†	-0.045***
# of writer patches	-0.012***	·	·	†

Does Participation in Code Review Influence Its Quality?

	Mozilla	Core	Firefox	Firefox for Android
Adjusted R^2	0.134	0.128	0.173	0.147
Size (LOC)	0.105***	0.103***	0.105***	0.117***
Chunks	†	†	†	†
Number of files	0.060***	0.059***	0.090***	0.067*
Module	⋆	n/a	n/a	n/a
Priority	‡	⋆	‡	⋆
Severity	⋆	‡	⋆	‡
Super review	-0.124**	-0.160***	‡	n/a
# of devs on CC	0.053***	0.056***	‡	0.049*
# of comments	†	†	†	†
# of commenting devs	-0.124***	-0.102***	-0.075***	-0.176***
# of comments / # devs	-0.039***	-0.029**	‡	‡
# of reviewer comments	0.010**	‡	0.026*	‡
# of writer comments	·	·	·	-0.047**

Study I: Insights

Investigating Code Review Quality:
Do People and Participation Matter?

Oleksii Kononenko*, Olga Baysal†, Latifa Guerrouj‡, Yaxin Cao†, and Michael W. Godfrey*
*David R. Cheriton School of Computer Science, University of Waterloo, Canada
Email: okononen@uwaterloo.ca, migod@uwaterloo.ca
† Department of Computer Science and Operations Research, Université de Montréal, Canada
Email: yaxin.cao@umontreal.ca, olga.baysal@umontreal.ca
‡ Département de Génie Logiciel et des Technologies de l'Information, École de technologie supérieure, Canada
Email: latifa.guerrouj@etsmtl.ca

Abstract—Code review is an essential element of any mature software development project; it aims at evaluating code contributions submitted by developers. In principle, code review should improve the quality of code changes (patches) before they are committed to the project's master repository. In practice, bugs are sometimes unwittingly introduced during this process.

In this paper, we report on an empirical study investigating code review quality for Mozilla, a large open-source project. We explore the relationships between the reviewers' code inspections and a set of factors, both personal and social in nature, that might affect the quality of such inspections. We applied the SZZ algorithm to detect bug-inducing changes that were then linked to the code review information extracted from the issue tracking system. We found that 54% of the reviewed changes introduced bugs in the code. Our findings also showed that both personal metrics, such as reviewer workload and experience, and participation metrics, such as the number of involved developers, are associated with the quality of the code review process.

Index Terms—Code review, code review quality, bug-inducing changes, mining software repositories, Mozilla, empirical study.

I. Introduction

Code review is an essential element of any mature software development project; it aims at evaluating code contributions submitted by developers. Code review is considered to be one of the most effective QA practices in software projects; it is relatively expensive in terms of time and effort, but provides good value in identifying defects in code changes before they are committed into the project's code base [1]. Reviewers are the gatekeepers of a project's master repository; they must carefully validate the design and implementation of patches to ensure that they meet expected quality standards.

In principle, the code review process should improve the quality of code changes (patches) before they are committed to the project's master repository. However, in practice, the execution of this process can still allow bugs to enter into the codebase unnoticed. This work aims to investigate the quality of code review.

In this paper, we have studied code review of a large open source system: the Mozilla project. For Mozilla, code review is an important and vital part of their development process, since contributions may come not only from core Mozilla developers but also from the greater user community. The Mozilla community embraces code review to help maintain consistent design and implementation practices among the many casual contributors and owners of various modules that comprise the Mozilla codebase [2]. They have found that code review improves code quality, promotes best practices, and reduces regressions [3].

The Mozilla code review process requires that every submitted patch be evaluated by at least one reviewer [4]. Reviewers are tasked to grant approval to a patch if 1) they believe that the patch is free of obvious bugs, and 2) the patch has test coverage appropriate to the change. If a reviewer feels unable to provide a timely, expert review on a certain patch, they can re-direct the patch to other reviewers who may be better able to perform the task. However, even given the careful scrutiny that patches undergo, software defects are still found after the changes have been reviewed and committed to the version control repository. These post-release defects raise red flags about the quality of code reviews. Poor-quality reviews that permit bugs to sneak in unnoticed can introduce stability, reliability, and security problems, affecting the user's experience and ultimately their satisfaction with the product.

Previous research on code review has examined topics such as the relation between code coverage/participation and software/design quality [5], [6]; however, the topic of code review quality remains largely unexplored. In this paper, we perform an empirical case study of a large open source Mozilla project including its top three largest modules: *Core*, *Firefox*, and *Firefox for Android*. We apply the SZZ algorithm [7] to detect bug-inducing changes that are then linked to the code review data extracted from the issue tracking system. We address the following overarching research questions:

RQ1 *Do code reviewers miss many bugs?*
The goal of code review is to identify problems (e.g., the code-level problems) in the proposed code changes. Yet, software systems remain bug-prone.

RQ2 *Do personal factors affect the quality of code reviews?*
Previous studies found that code ownership has a strong relationship with both pre- and post-release defects-proneness [8]–[10].

RQ3 *Does participation in code review influence its quality?*
A recent study demonstrated that low review participation has a negative impact on software quality [5].

978-1-4673-7532-0/15/$31.00 @ 2015 IEEE

Practice

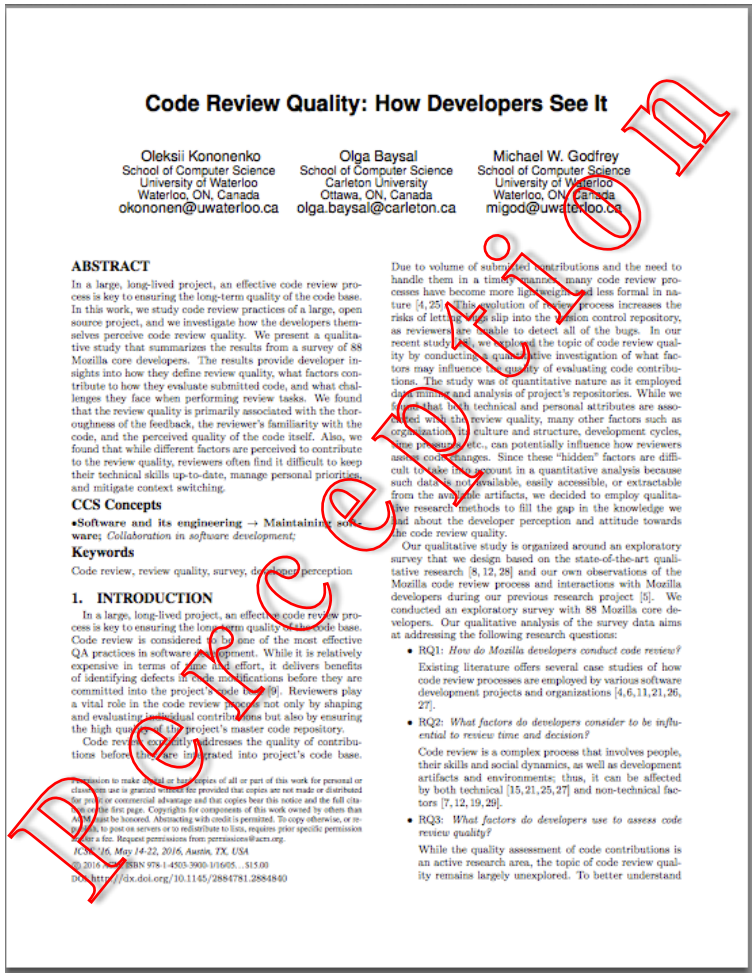
Kononenko et al., ICSME 2015

Reviewers miss many problems

Review quality factors:

- Size-related metrics
- Participation in discussion of bug fixes
- Reviewer workload
- Reviewer experience

Study II: Developer Perception



Code Review Quality: How Developers See It

Oleksii Kononenko
School of Computer Science
University of Waterloo
Waterloo, ON, Canada
okononen@uwaterloo.ca

Olga Baysal
School of Computer Science
Carleton University
Ottawa, ON, Canada
olga.baysal@carleton.ca

Michael W. Godfrey
School of Computer Science
University of Waterloo
Waterloo, ON, Canada
migod@uwaterloo.ca

ABSTRACT

In a large, long-lived project, an effective code review process is key to ensuring the long-term quality of the code base. In this work, we study code review practices of a large, open source project, and we investigate how the developers themselves perceive code review quality. We present a qualitative study that summarizes the results from a survey of 88 Mozilla core developers. The results provide developer insights into how they define review quality, what factors contribute to how they evaluate submitted code, and what challenges they face when performing review tasks. We found that the review quality is primarily associated with the thoroughness of the feedback, the reviewer's familiarity with the code, and the perceived quality of the code itself. Also, we found that while different factors are perceived to contribute to the review quality, reviewers often find it difficult to keep their technical skills up-to-date, manage personal priorities, and mitigate context switching.

CCS Concepts

•**Software and its engineering** → **Maintaining software;** *Collaboration in software development;*

Keywords

Code review, review quality, survey, developer perception

1. INTRODUCTION

In a large, long-lived project, an effective code review process is key to ensuring the long-term quality of the code base. Code review is considered to be one of the most effective QA practices in software development. While it is relatively expensive in terms of time and effort, it delivers benefits of identifying defects in code modifications before they are committed into the project's code base [9]. Reviewers play a vital role in the code review process not only by shaping and evaluating individual contributions but also by ensuring the high quality of the project's master code repository.

Code review explicitly addresses the quality of contributions before they are integrated into project's code base. Due to volume of submitted contributions and the need to handle them in a timely manner, many code review processes have become more lightweight and less formal in nature [4, 25]. This evolution of review process increases the risks of letting bugs slip into the version control repository, as reviewers are unable to detect all of the bugs. In our recent study [28], we explored the topic of code review quality by conducting a quantitative investigation of what factors may influence the quality of evaluating code contributions. The study was of quantitative nature as it employed data mining and analysis of project's repositories. While we found that both technical and personal attributes are associated with the review quality, many other factors such as organizational culture and structure, development cycles, time pressure, etc., can potentially influence how reviewers assess code changes. Since these "hidden" factors are difficult to take into account in a quantitative analysis because such data is not available, easily accessible, or extractable from the available artifacts, we decided to employ qualitative research methods to fill the gap in the knowledge we had about the developer perception and attitude towards the code review quality.

Our qualitative study is organized around an exploratory survey that we design based on the state-of-the-art qualitative research [8, 12, 28] and our own observations of the Mozilla code review process and interactions with Mozilla developers during our previous research project [5]. We conducted an exploratory survey with 88 Mozilla core developers. Our qualitative analysis of the survey data aims at addressing the following research questions:

- RQ1: *How do Mozilla developers conduct code review?*
  Existing literature offers several case studies of how code review processes are employed by various software development projects and organizations [4, 6, 11, 21, 26, 27].
- RQ2: *What factors do developers consider to be influential to review time and decision?*
  Code review is a complex process that involves people, their skills and social dynamics, as well as development artifacts and environments; thus, it can be affected by both technical [15, 21, 25, 27] and non-technical factors [7, 12, 19, 29].
- RQ3: *What factors do developers use to assess code review quality?*
  While the quality assessment of code contributions is an active research area, the topic of code review quality remains largely unexplored. To better understand

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICSE '16, May 14-22, 2016, Austin, TX, USA
© 2016 ACM. ISBN 978-1-4503-3900-1/16/05…$15.00
DOI: http://dx.doi.org/10.1145/2884781.2884840

Preview

We conducted a survey with Mozilla developers and asked them:

- How do you conduct code reviews?
- **What influences review time and decision?**
- **What is a good review?**
- What challenges do you face?

Kononenko et al., ICSE 2016

Survey Participants

Bugzilla
(12 months period)

88 responses
(22% response rate)

74% : ≥ 7 years of programming exp.

67% : ≥ 3 years of code review exp.

3,142 contributors

843 developers
with exp. value ≥ 15

403 “mature” developers

Survey Analysis

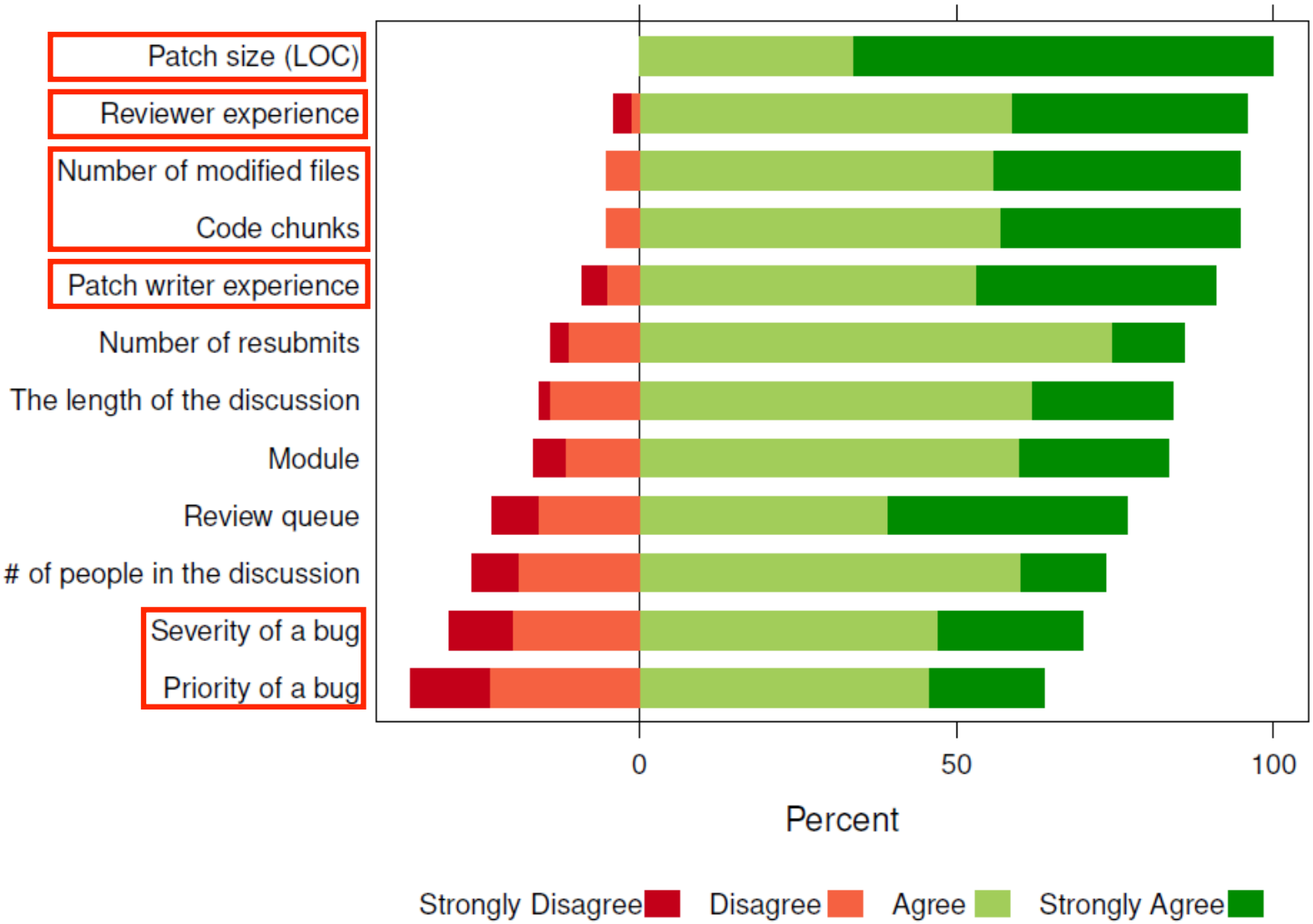
12 closed-ended questions

Survey data

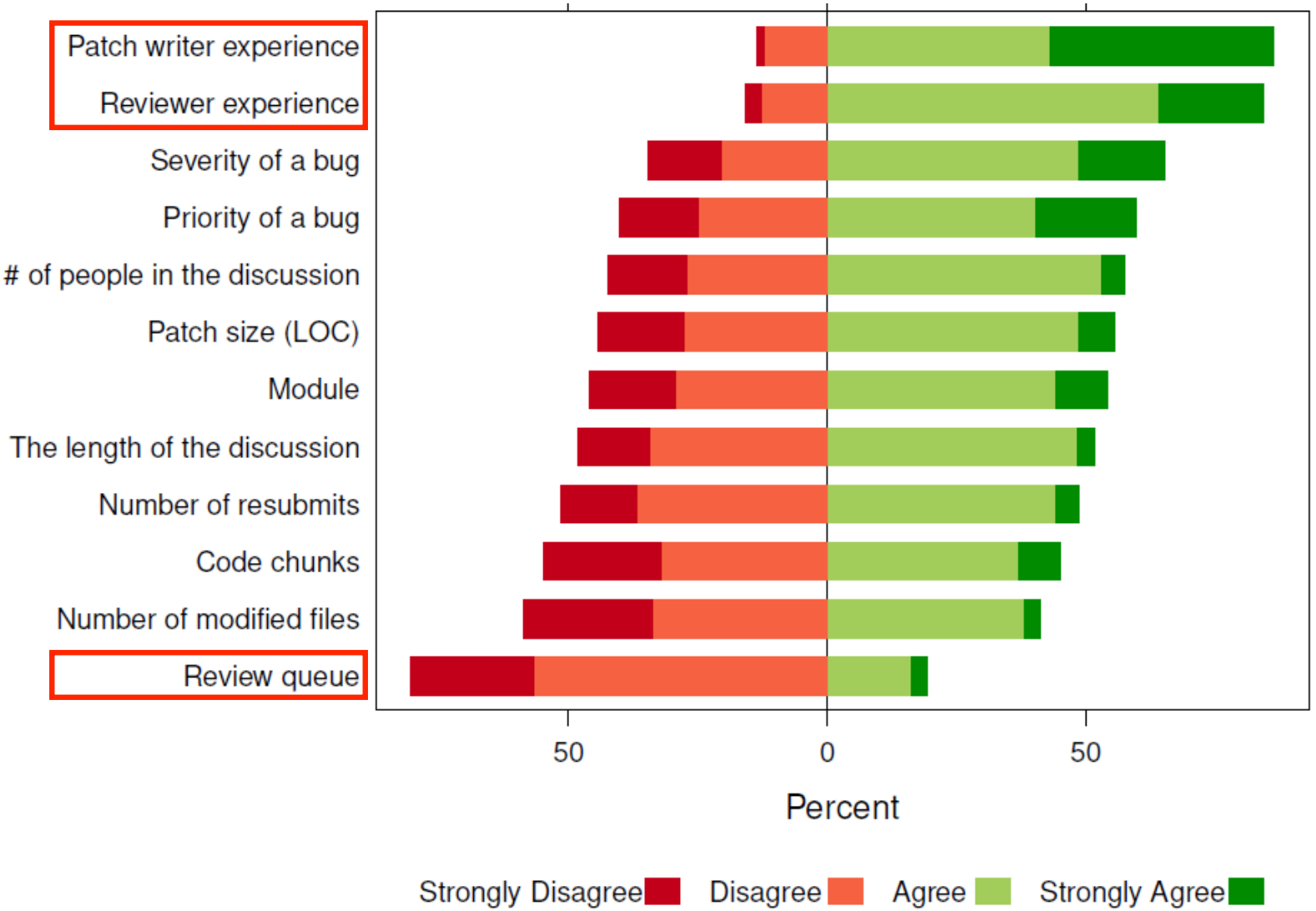
7 open-ended questions

Card sorting
(978 quotes, 30 categories)

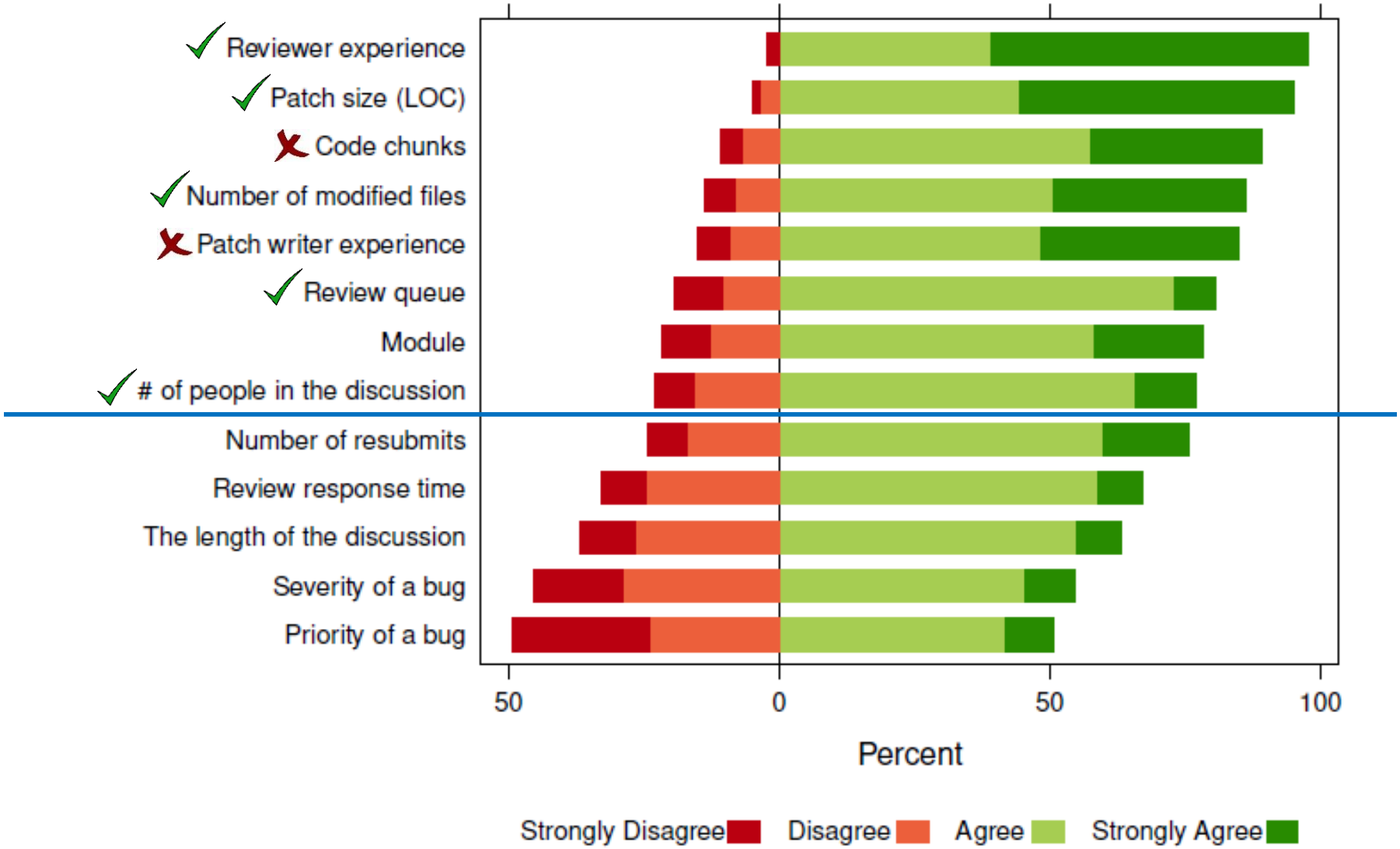
Factors Influencing CR Time



Factors Influencing CR Decision



Factors Affecting CR Quality



Study II: Insights

Mozilla developers believe that:

- Patch size, reviewer experience, and tests do matter
- Reviewers should be experts

Code quality matters, but so do these:

- High quality feedback
- Reviewer's personal qualities
- Trust and familiarity with the patch author

Code Review in ML Libraries

Investigating Code Review Quality in ML Libraries

Vivek Thaker[1*†], Michael Godfrey[2] and Olga Baysal[1†]

[1]School of Computer Science, Carleton University, 1125 Colonel By Drive, Ottawa, K1S 5B6, Ontario, Canada.
[2]Cheriton School of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo, N2L 3G1, Ontario, Canada.

*Corresponding author(s). E-mail(s): vivekthaker@cmail.carleton.ca;
Contributing authors: migod@uwaterloo.ca; olga.baysal@carleton.ca;
†These authors contributed equally to this work.

Abstract

Over the past several years, ML techniques have become commonplace in numerous technological areas where many real-world environments depend on such techniques. More recently, tasks that relied on traditional ML approaches are being replaced or incorporated alongside more modern, complex techniques such as large language models (LLMs). In the existing ML landscape, open-source codebases are often used regularly by a multitude of groups and organizations of all sizes. As a result of their pervasiveness, bugs in such codebases may have significant, far-reaching consequences. While developers of such large-scale codebases typically strive to minimize bugs and ensure high-quality code is deployed, bugs still exist in production-level code and are typically more difficult to detect/resolve than non-ML codebases due to the greater complexity of these codebases.

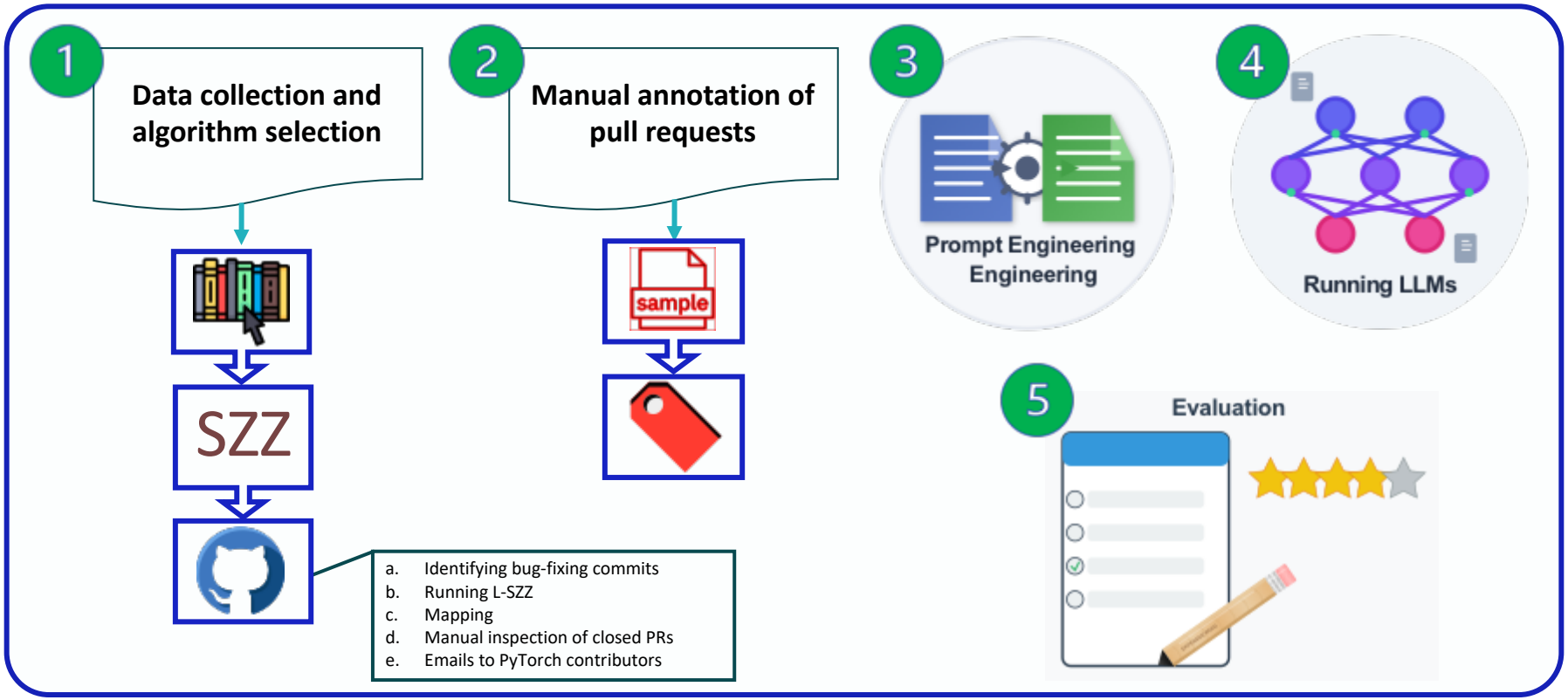
The extent to which bugs reach production code depends on the quality of the written code as well as the quality of the code review routines employed by the maintainers of the open-source ML codebases. While there is a wealth of literature that inspects many aspects related to code review, bugs in ML applications, the general use of LLMs, etc., there is non-existent literature that explores the quality of code review particularly in popular open-source ML libraries as well as the role that LLMs play in automated detection of bug types in such libraries. The goal of this thesis is to address the knowledge gap regarding the quality of code review in open-source ML libraries and evaluate the capabilities of state-of-the-art LLMs in detecting the types of bugs during the code review phase. In this thesis, we inspect five of the most modern, open-source, and popular ML

1

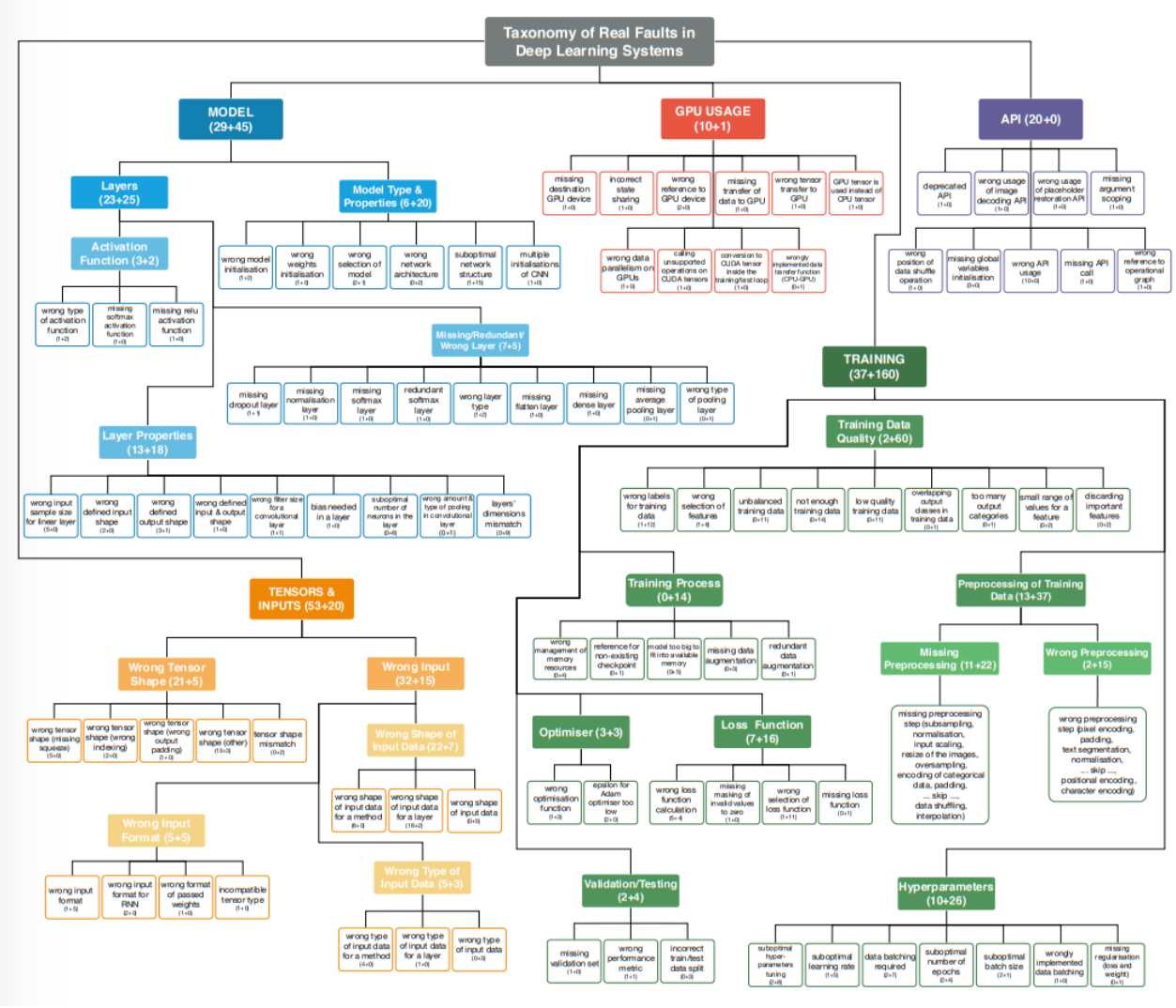
EMSE 2025 (In preparation)

- Mining GitHub PRs
- **Do developers miss many bugs?**
- **What types of bugs are typically missed?**

Methodology



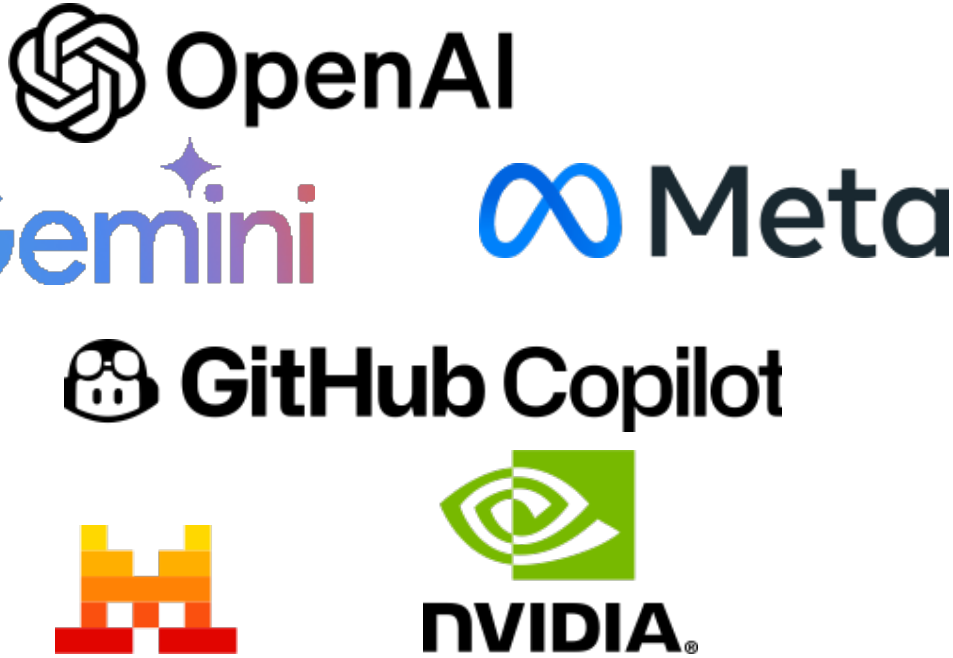
Taxonomy of DL Faults



Nargiz Humbatova, Gunel Jahangirova, Gabriele Bavota, Vincenzo Riccio, Andrea Stocco, Paolo Tonella. “Taxonomy of Real Faults in Deep Learning Systems”. ICSE 2020.

LLMs

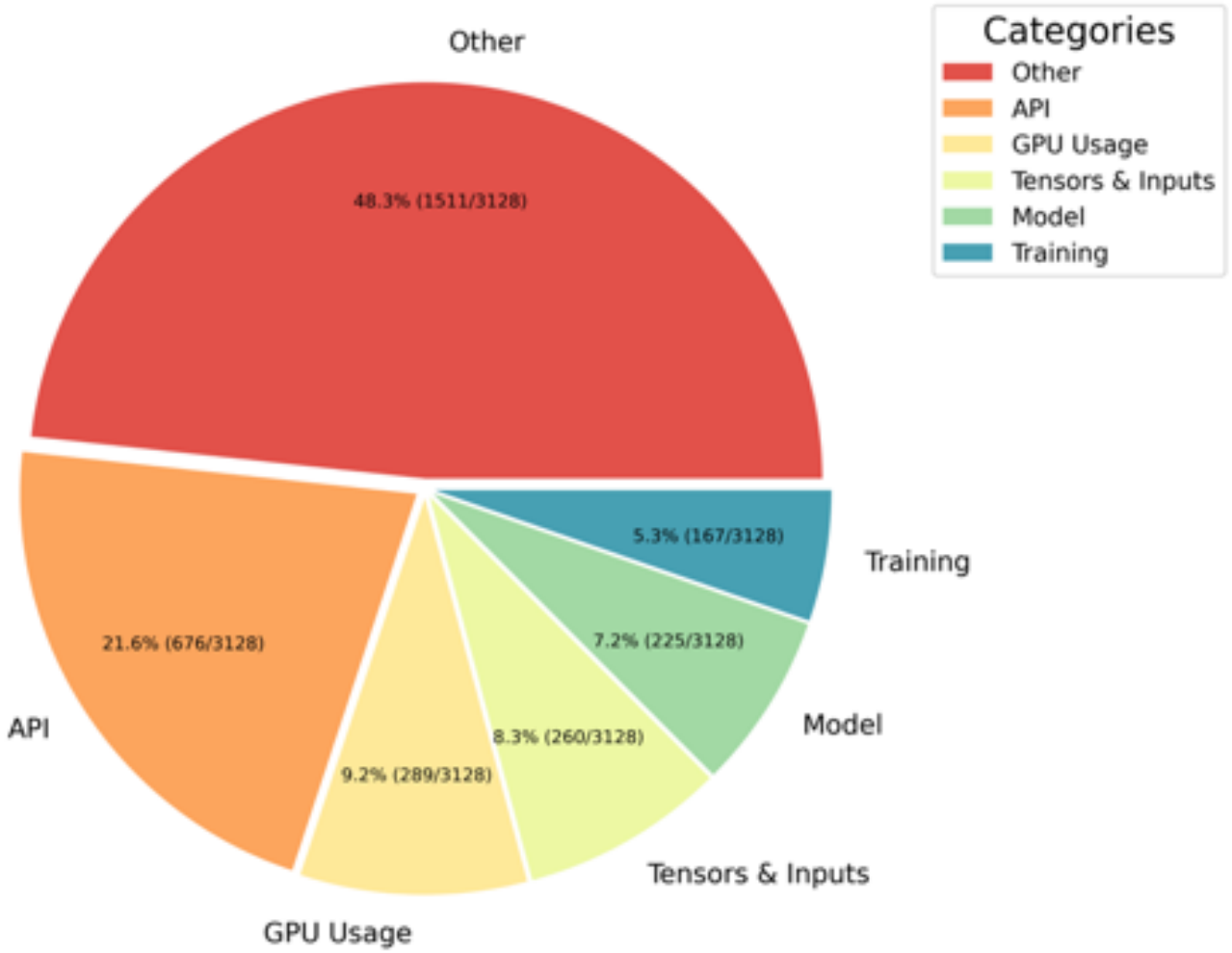
- **OpenaAI**
  - GPT 4o-mini
  - GPT 4o
- **Meta**
  - Llama 3.2 1B-Instruct
  - Llama 3.2 3B-Instruct
  - Llama 3.1 8B-Instruct
- **GitHub Copilot**
- **Google Gemini**
  - 1.5-Flash
  - 1.5-Flash 8B
- **Nvidia**
  - Llama 3.1 Nemotron 70B-Instruct
- **Mistral AI**
  - Ministral 8B



Do Reviewers Miss Many Bugs?

Project	Number of bug-inducing PRs	Number of merged PRs	%
Pandas	1,236	10,557	11.70%
Scikit-learn	517	4,303	12.01%
TensorFlow	811	5,869	13.81%
NumPy	637	3,752	16.97%
PyTorch	314	1,224	25.65%

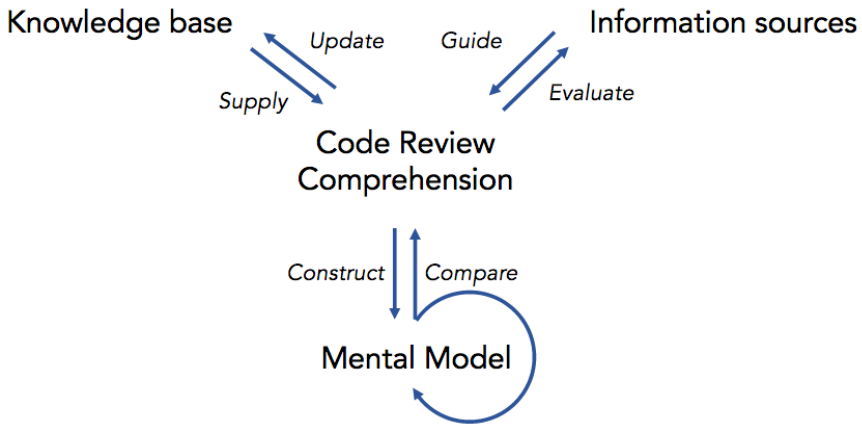
What Types of Bugs Are Missed?



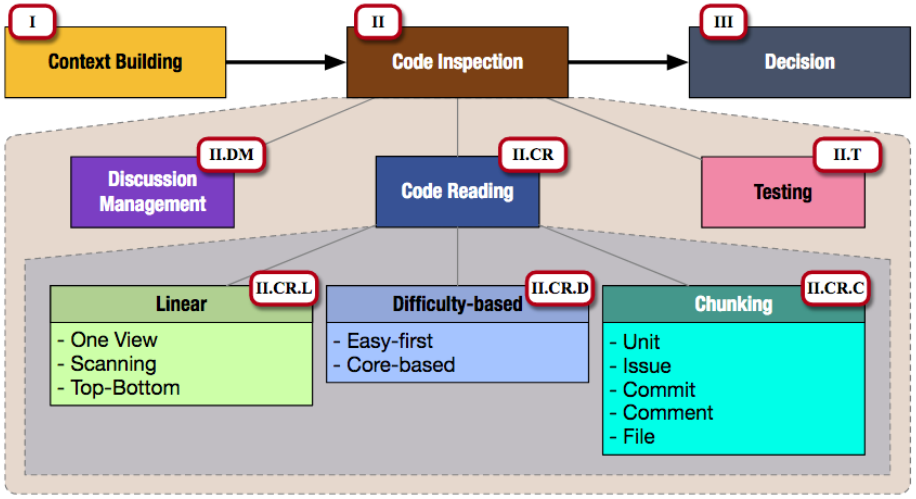
Study III: Insights

- Code review quality of ML libraries is relatively high
- Code review quality is project-dependent
- Pandas had fewest missed bugs, PyTorch the most
- LLMs are sensitive to the bug type

Other Theories of Code Review?



Code Review Comprehension Model



Activities for Understanding and Creating Review Strategy

Pavlina Wurzel Goncalves, Pooja Rani, Margaret-Anne Storey, Diomidis Spinellis, Alberto Bacchelli “Code Review Comprehension: Reviewing Strategies Seen Through Code Comprehension Theories”. ICPC 2025.

Can AI-Powered Assistants Improve Code Review Quality?

AI in Software Development

AI in the development workflow

Developers currently using AI tools mostly use them to write code (82%) and those who are interested but not yet using AI tools are mostly curious about testing code (46%). Developers with experience can trust AI tools to help write code to get started but perhaps know testing is a complex step best left to traditional processes.

Which parts of your development workflow are you currently using AI tools for and which are you interested in using AI tools for over the next year? Please select all that apply.

2024 Developer Survey

Source: survey.stackoverflow.co/2024
Data licensed under Open Database License (ODbL)

https://survey.stackoverflow.co/2024/ai

AI in Software Development

AI in the development workflow

Developers currently using AI tools mostly use them to write code (82%) and those who are interested but not yet using AI tools are mostly curious about testing code (46%). Developers with experience can trust AI tools to help write code to get started but perhaps know testing is a complex step best left to traditional processes.

Which parts of your development workflow are you currently using AI tools for and which are you interested in using AI tools for over the next year? Please select all that apply.

2024 Developer Survey

Source: survey.stackoverflow.co/2024
Data licensed under Open Database License (ODbL)

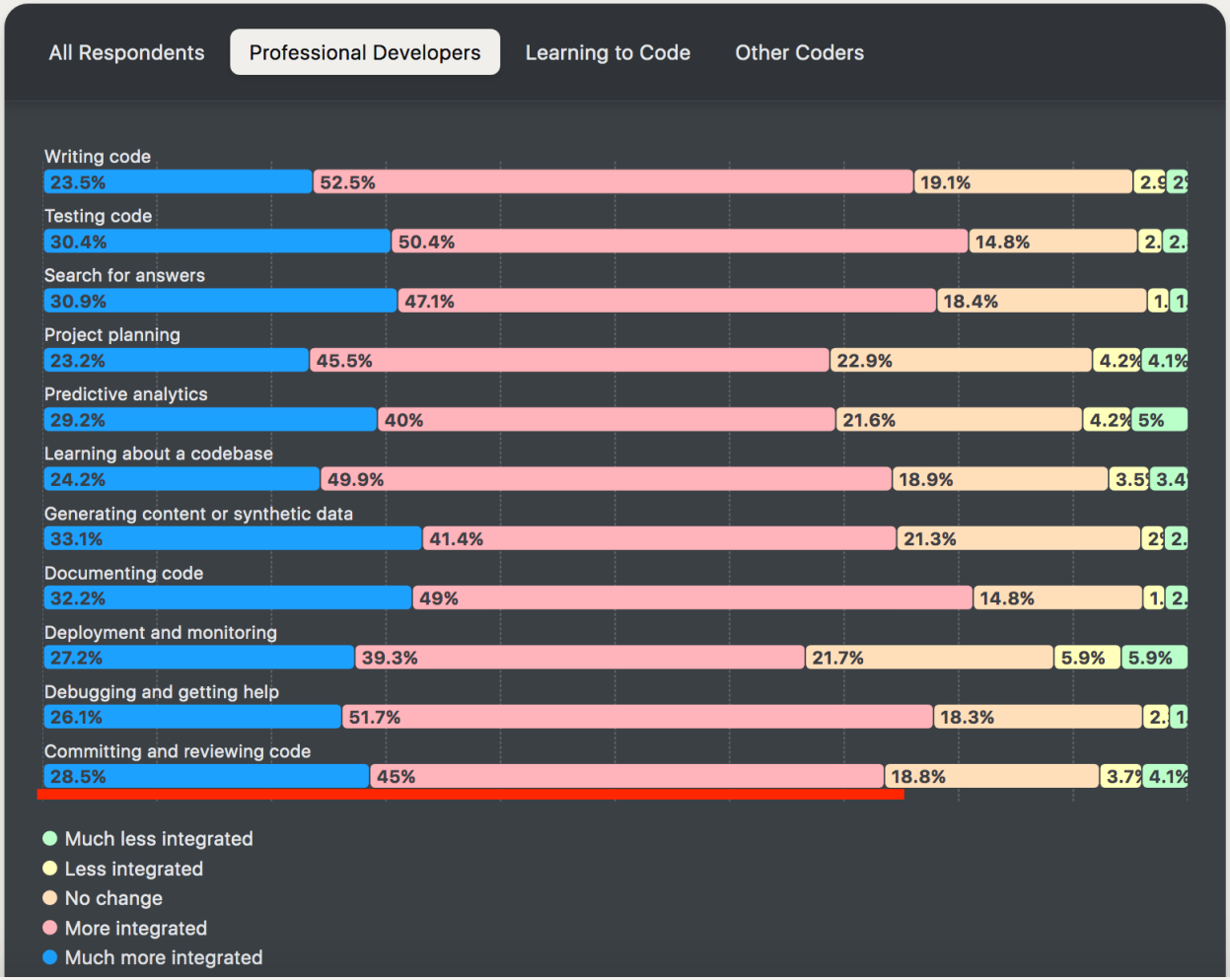
https://survey.stackoverflow.co/2024/ai

AI in Software Development

AI tools next year

In the next year, most developers agree that AI tools will be more integrated mostly in the ways they are documenting code (81%), testing code (80%), and writing code (76%).

Thinking about how your job and process changes over time, how integrated in your workflow do you anticipate AI tools you are currently using will be 1 year from now?



2024 Developer Survey

Source: survey.stackoverflow.co/2024

Data licensed under Open Database License (ODbL)

https://survey.stackoverflow.co/2024/ai

Examples

- **GitHub Copilot + PR Reviews**
  AI can suggest improvements, detect code smells, or even offer inline explanations of complex logic during the review process

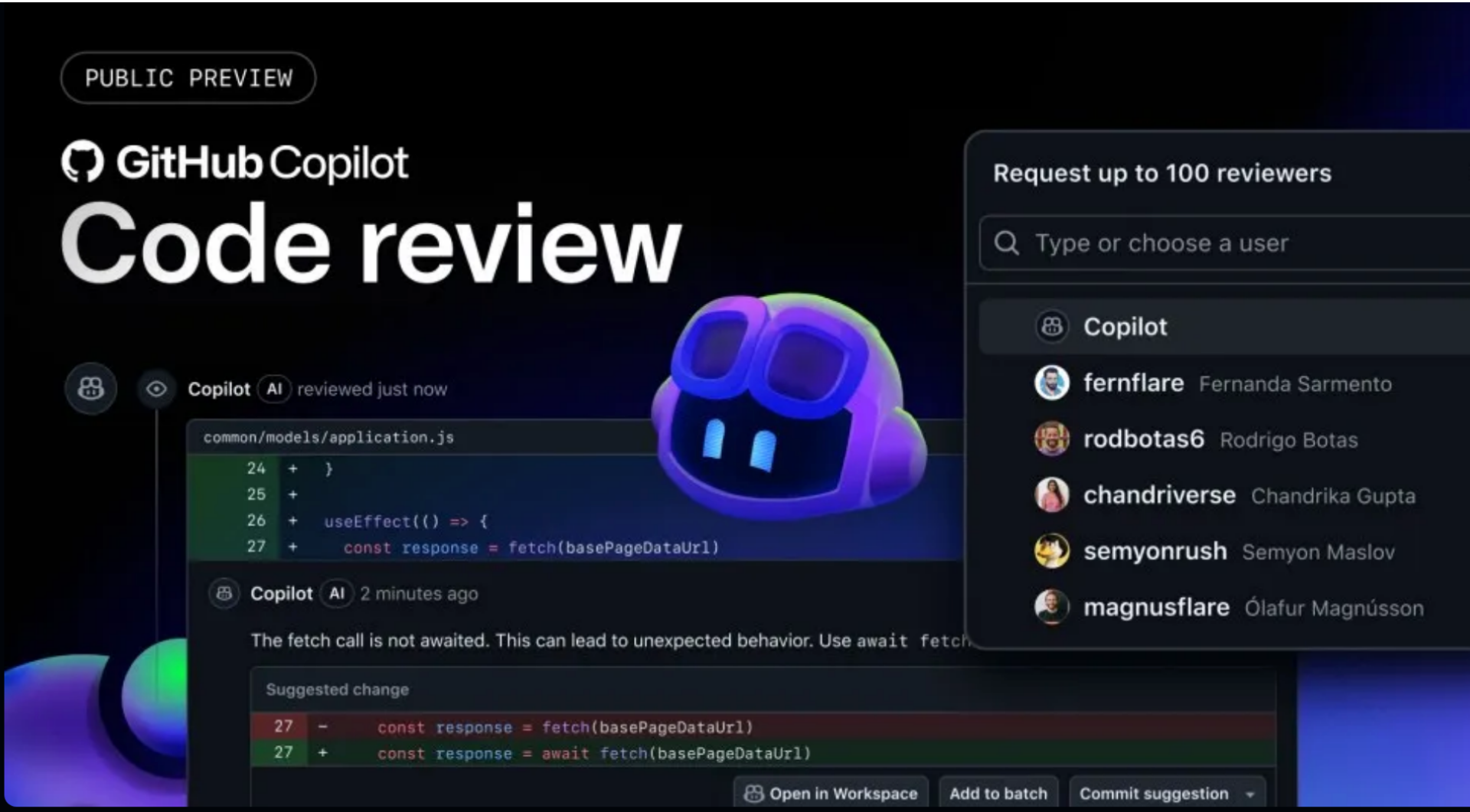
- **DeepCode / Snyk Code**
  AI-driven static analysis tools like Snyk Code to automatically flag vulnerabilities or bad practices during review.

- **Meta: Internal AI for Code Review Prioritization**
  ML models to predict which pull requests are likely to have defects

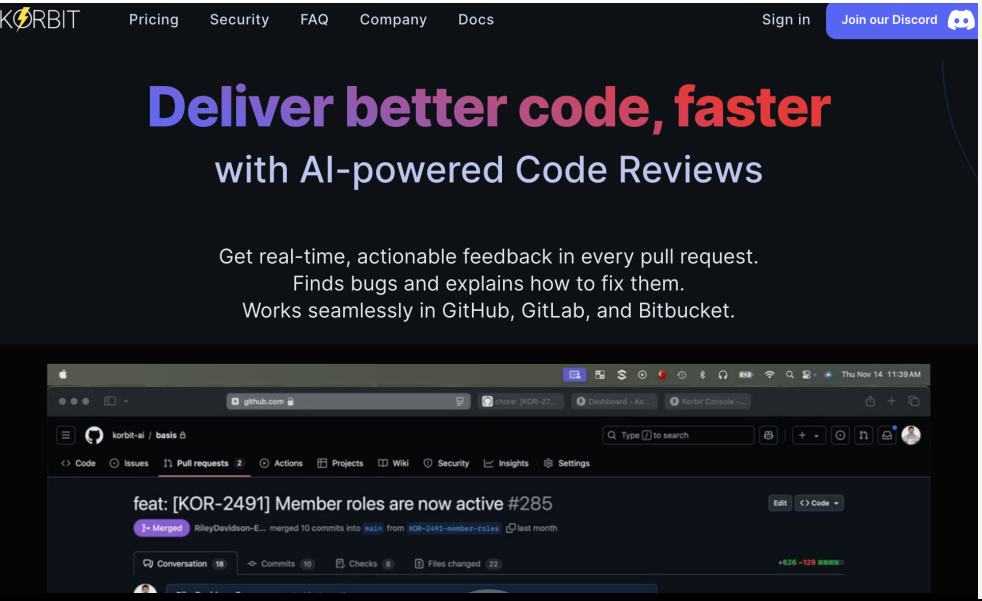
- **Google: Automated Suggestions via Code Review Bots**
  Tools Critique and Tricorder to automatically suggest style, formatting, or even logic improvements in PRs.

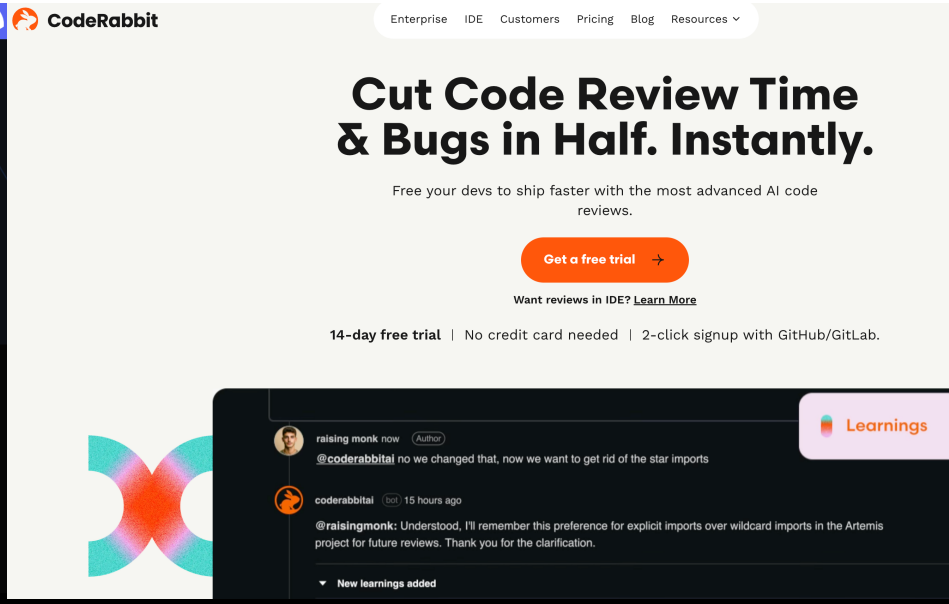
- **Amazon: AI-Assisted Reviewer Matching**

Examples: GitHub Copilot

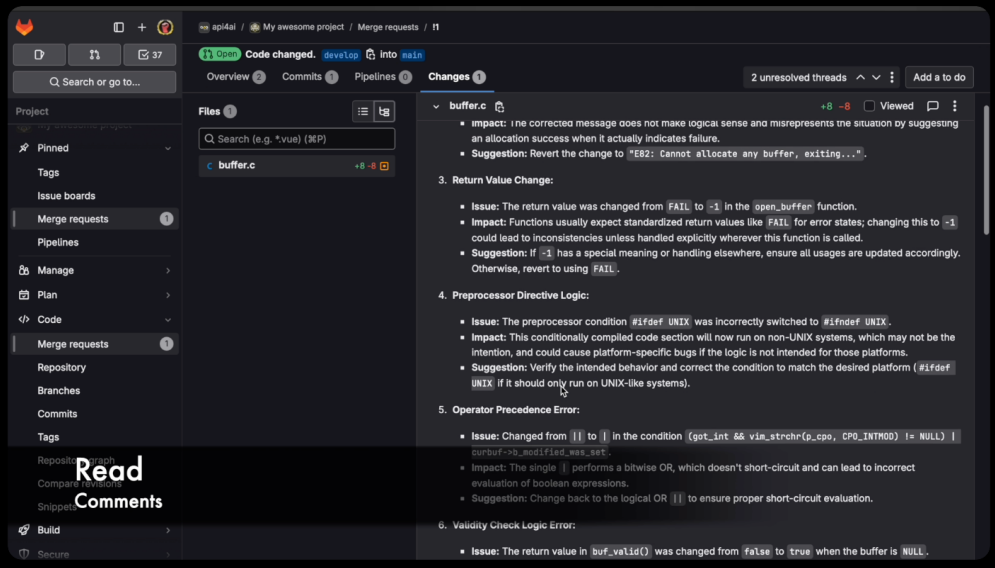


Examples


KORBIT
Pricing
Security
FAQ
Company
Docs
Sign in
Join our Discord
Deliver better code, faster
with AI-powered Code Reviews
Get real-time, actionable feedback in every pull request.
Finds bugs and explains how to fix them.
Works seamlessly in GitHub, GitLab, and Bitbucket.
feat: [KOR-2491] Member roles are now active #285


CodeRabbit
Enterprise
IDE
Customers
Pricing
Blog
Resources
Cut Code Review Time
& Bugs in Half. Instantly.
Free your devs to ship faster with the most advanced AI code reviews.
Get a free trial
Want reviews in IDE? Learn More
14-day free trial | No credit card needed | 2-click signup with GitHub/GitLab.
Learnings
raising monk now Author
@coderabbitai no we changed that, now we want to get rid of the star imports
coderabbitai bot 15 hours ago
@raisingmonk: Understood, I'll remember this preference for explicit imports over wildcard imports in the Artemis project for future reviews. Thank you for the clarification.
New learnings added


CRken
Your Ultimate AI Assistant for
Automating Code Review
Inspects code for potential bugs
Seamless integration with GitLab
Free 1-month trial (no credit card)


Code changed.
Overview
Commits
Pipelines
Changes
2 unresolved threads
Add a to do
buffer.c
Viewed
Read
Comments

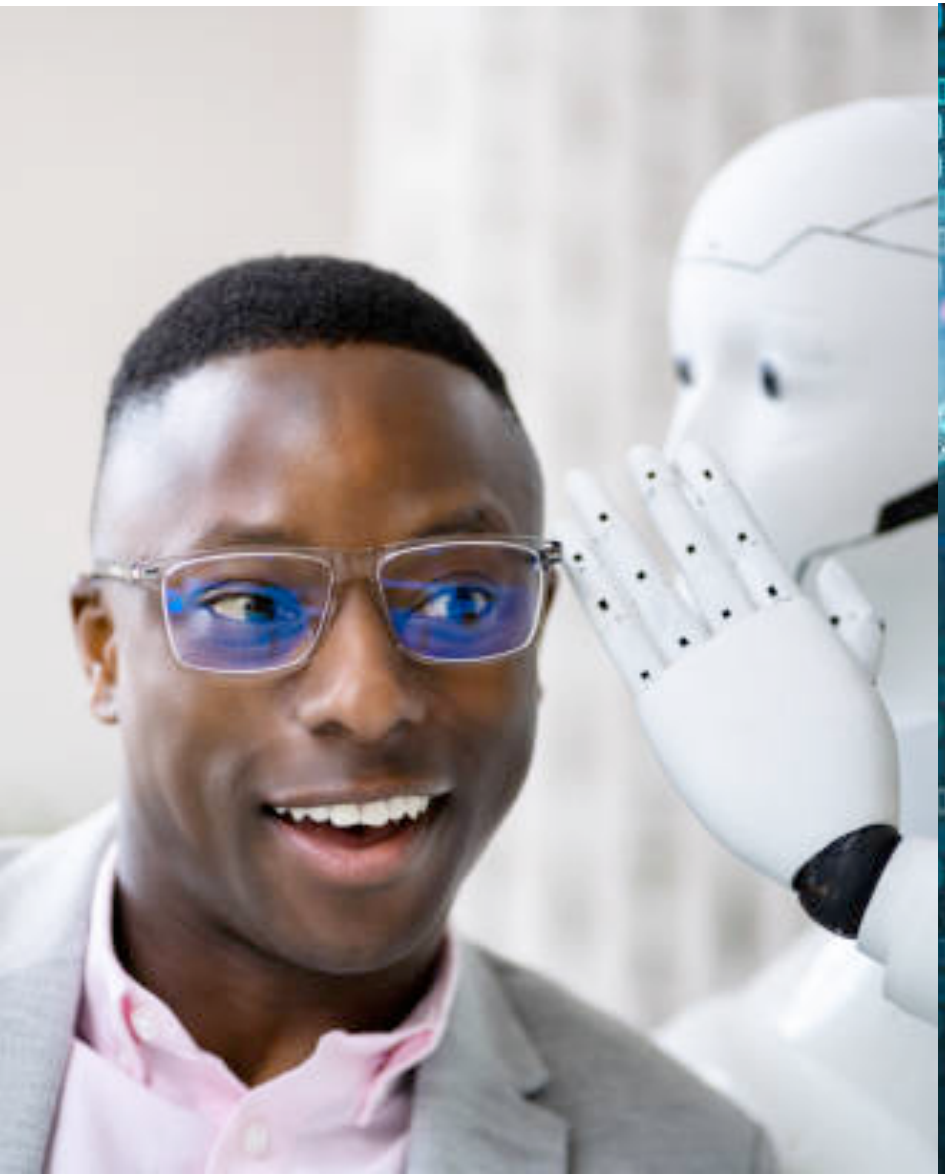
What AI Does Well

- Automates formatting/ style checks
- Suggests fixes, documents code, summarizes changes
- Recommends code reviewers

What Humans (Will) Do Best

- Understand context and business logic
- Make architectural decisions
- Mentor and guide team growth

(Some) AI Side Effects

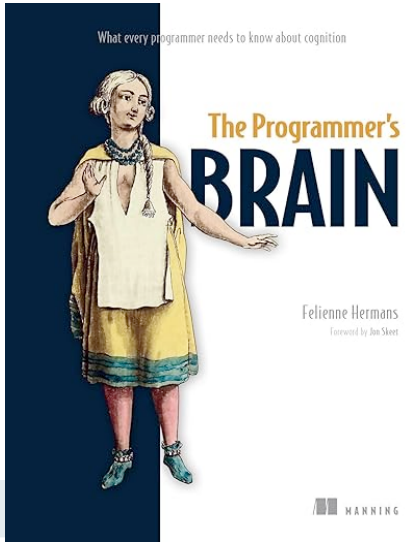


AI Overreliance

AI killed my coding brain but I'm rebuilding it

We sprinted into the AI age of autocomplete IDEs now we're waking up wondering why we forgot how to write a for-loop.

Devlink Tips Follow 16 min read · May 11, 2025



reddit

r/csMajors Search in r/csMajors

Home
Popular
Answers BETA

TOPICS

Internet Culture (Viral)
Games
Q&As
Technology

r/csMajors • 4 mo. ago
CommentNo2882

AI Ruined My Ability to Learn Deeply / I need advice

I started learning how to code two years before AI tools became widely available. I was still in high school, and it was during COVID, so I didn't pay much attention and didn't learn how to code in depth. I was able to make some websites and code small projects by learning through YouTube videos. I stopped coding for a while, and when I came back, AI-assisted tools were everywhere. After that, I started using them daily for my learning and projects, which was the biggest mistake I ever made.

I can understand 90% of the code that is put in front of me (school work, not full codebases), and it's fine—I can read it. But now, if you ask me to write code from scratch using any of the languages I know, I simply can't. I have no memory of the syntax and no idea what the next step is, but when I read code, it feels so easy to understand. I know what to do, I know how to use pseudocode, but I simply cannot translate that into actual code in VSCode.

I feel like a fraud—I can read code (which is easy) but cannot write it. I struggle with projects.

Legal Issues and Regulations

We've filed a lawsuit challenging GitHub Copilot, an AI product that relies on unprecedented open-source software piracy.
Because AI needs to be fair & ethical for everyone.

NOVEMBER 3, 2022

Getty Images is suing the creators of AI art tool Stable Diffusion for scraping its content

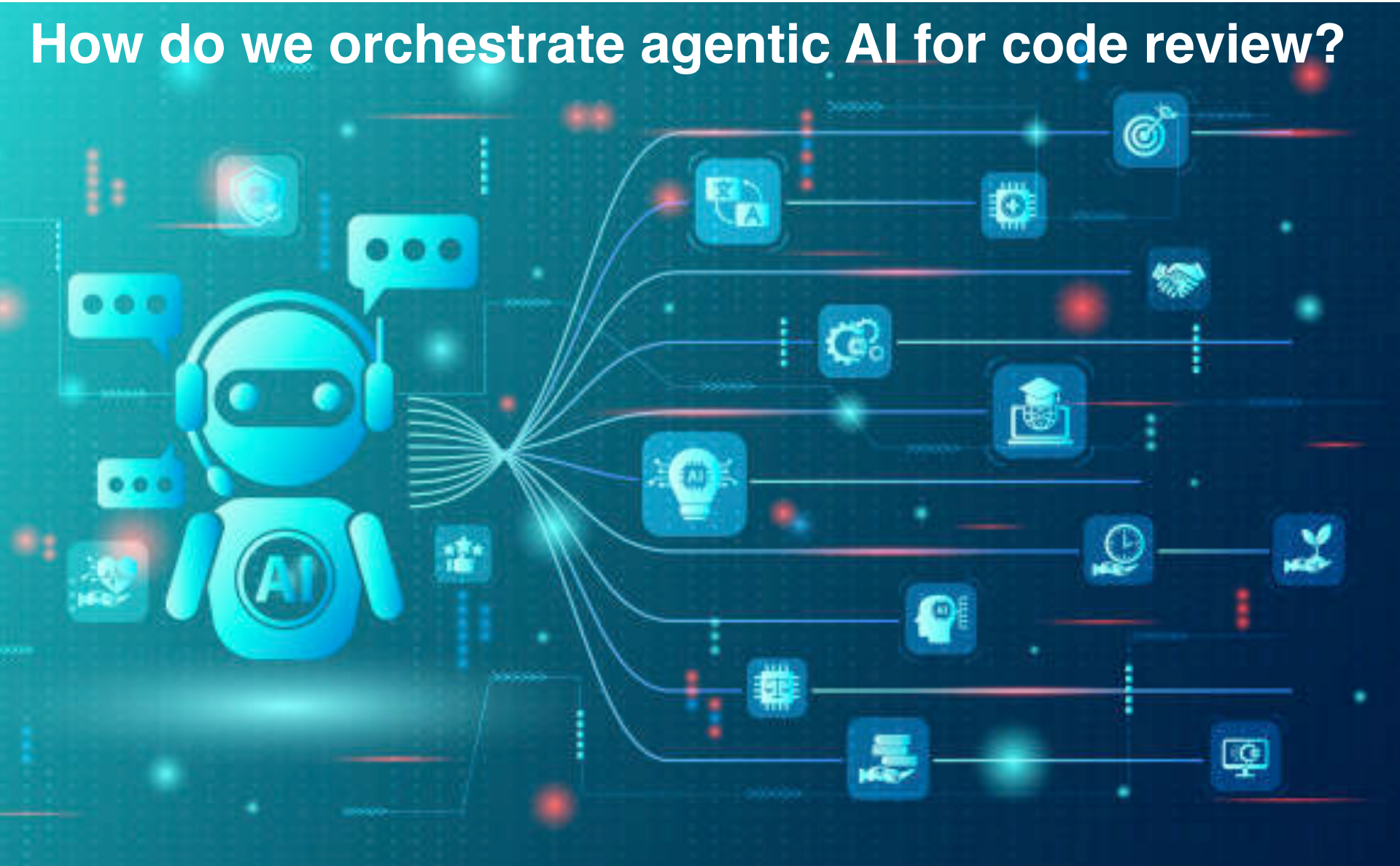
An image created by Stable Diffusion showing a recreation of Getty Images' watermark.
Image: The Verge / Stable Diffusion

/ Getty Images claims Stability AI 'unlawfully' scraped millions of images from its site. It's a significant escalation in the developing legal battles between generative AI firms and content creators.

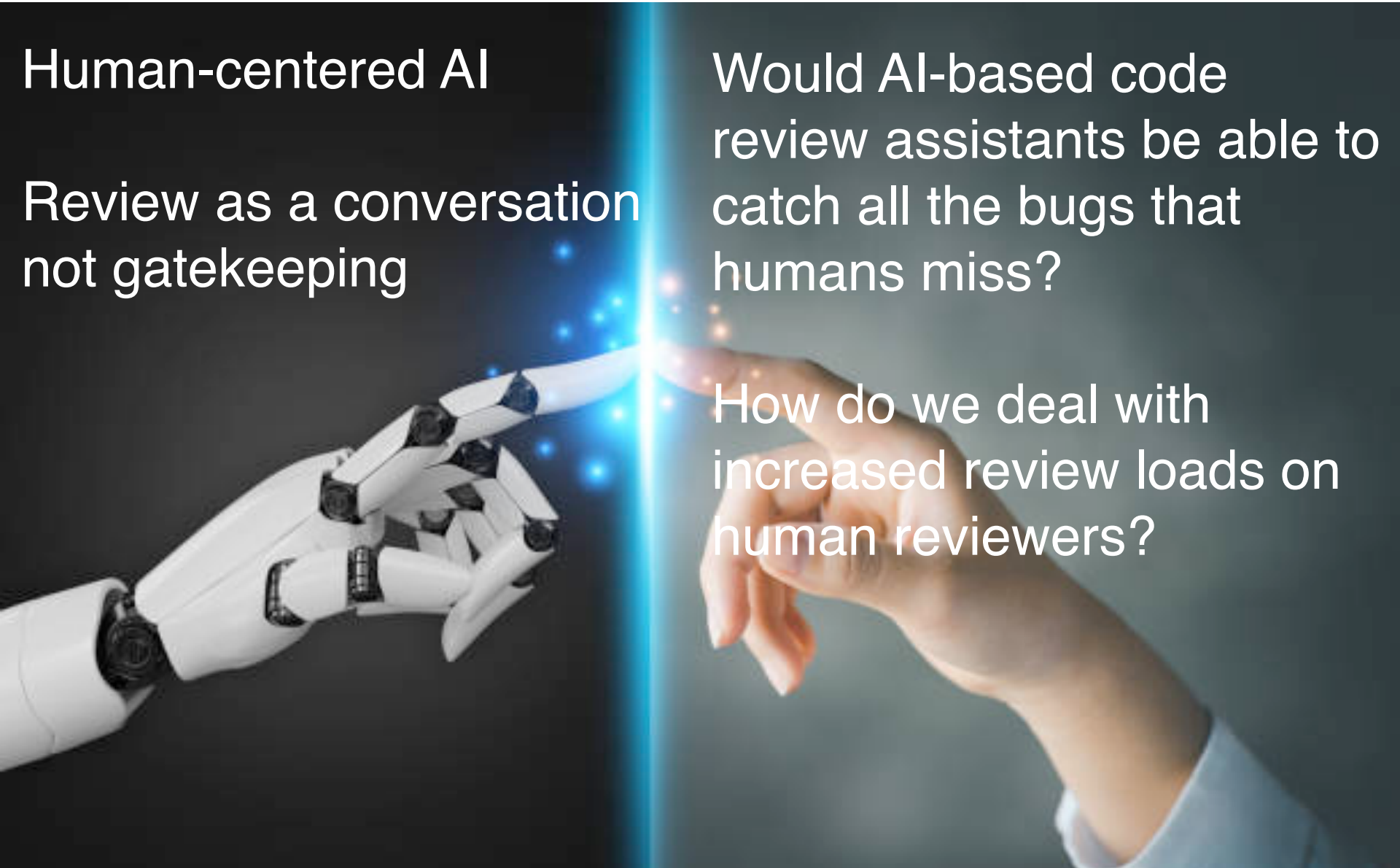
by James Vincent
Jan 17, 2023, 5:30 AM EST

18 Comments (18 New)

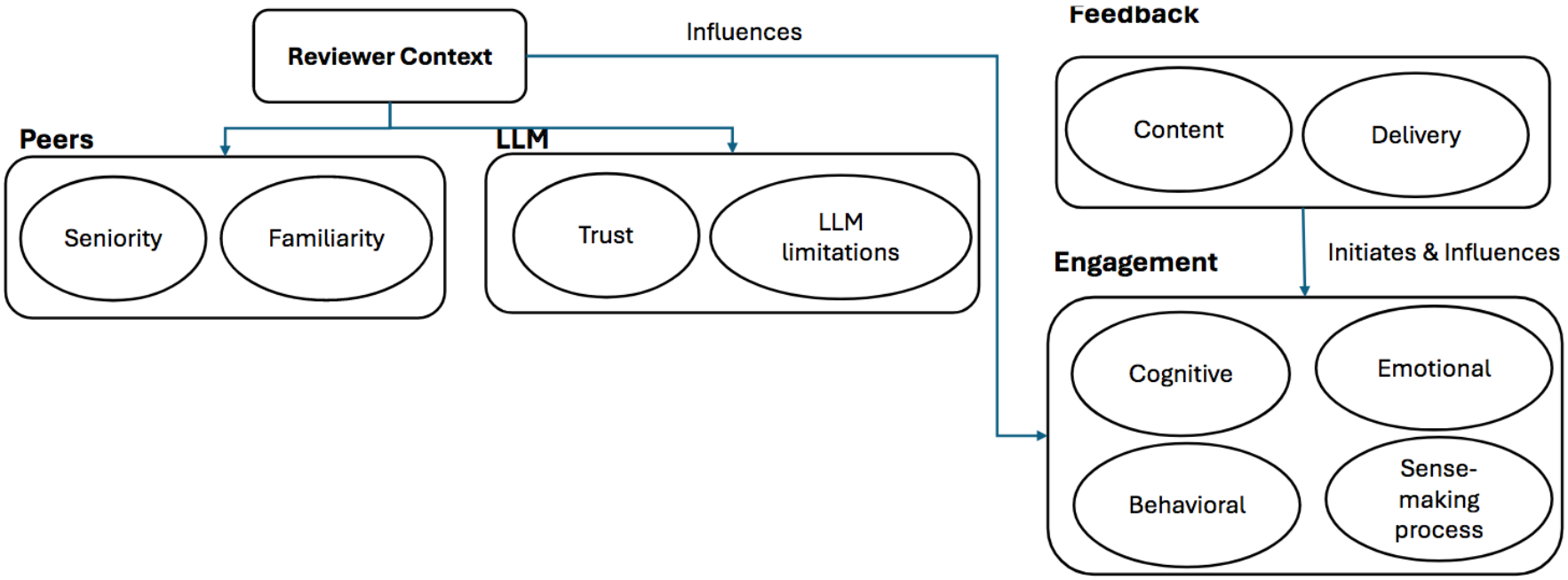
Opportunities



Opportunities



Developer Engagement with AI-Assisted Code Reviews



Adam Alami, Neil Ernst. “Human and Machine: How Software Engineers Perceive and Engage with AI-Assisted Code Reviews Compared to Their Peers”. CHASE 2025.

Opportunities

Building **ethical** AI-powered assistants:

- Security
- Accountability
- Data privacy

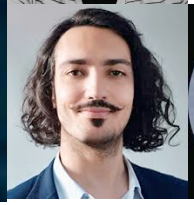
…

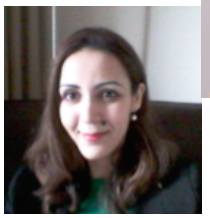
Developers make mistakes during code review

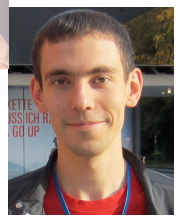
AI-powered tools help developers avoid mistakes during code review

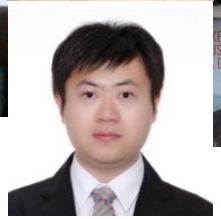
Developers make mistakes during code review

Can AI-Powered Assistants Improve Code Review Quality?









shopify Mozilla

Image Credits:
https://www.istockphoto.com