Understanding the Rejection of Fixes Generated by Agentic Pull Requests - Insights from the AIDev Dataset

Mahmoud Abujadallah
Software Engineering and IT
École de Technologie Supérieure
Montréal, Québec, Canada
mahmoud.abujadallah.1@ens.etsmtl.ca

Ali Arabat
Software Engineering and IT
École de Technologie Supérieure
Montréal, Québec, Canada
ali.arabat.1@ens.etsmtl.ca

Mohammed Sayagh
Software Engineering and IT
École de Technologie Supérieure
Montréal, Québec, Canada
Mohammed.Sayagh@etsmtl.ca

Abstract

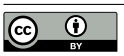
AI coding agents are increasingly used to generate pull requests (PRs) that propose code fixes in software projects. From a first exploration of the AIDev dataset, we find that 46.41% of the fixes proposed by the agents Copilot, Devin, Cursor, and Claude are rejected. This represents a significant amount of wasted resources that require human reviews, verifications, and running tests and validations for fixes that are merely discarded. Our goal in this paper is to understand the failure modes of AI-agents, an understanding that is crucial for better integrating AI-agents as efficient teammates. In this paper, we conduct a qualitative study on a representative sample of 306 non-merged pull requests created or co-authored by the agents mentioned earlier, followed by a quantitative analysis of the reasons for rejection. Our qualitative findings identify 14 reasons divided into four high-level categories for rejecting AI-agent fixes. We observe that developers can reject fixes due to fixes whose implementation is incorrect (e.g., incomplete, wrong approach), fixes that do not pass the continuous integration (CI) pipelines and fail tests, fixes for which the agent is unable to perform the implementation (e.g., no code generated, sessions lost), and fixes whose priority is low. Our results shed light on the importance of better guiding the model at these levels: (1) proposing hints about the approach to follow for fixing an issue, (2) outlining constraints or limitations regarding the approaches that should not be taken, and (3) instructing the agent on how to validate the implementation through CI pipelines and without introducing a breaking change. Our results suggest the need for good prioritization of tasks so that generated fixes do not lead to wasted human review efforts or wasted agent resources (e.g., tokens, compute, or allowed number of requests).

CCS Concepts

• **Software and its engineering → Collaboration in software development**.

Keywords

AI-agents, Agentic-PRs, Fixing issues, Qualitative study



This work is licensed under a Creative Commons Attribution 4.0 International License.
MSR '26, Rio de Janeiro, Brazil
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2474-9/26/04
https://doi.org/10.1145/3793302.3793592

ACM Reference Format:
Mahmoud Abujadallah, Ali Arabat, and Mohammed Sayagh. 2026. Understanding the Rejection of Fixes Generated by Agentic Pull Requests - Insights from the AIDev Dataset. In *23rd International Conference on Mining Software Repositories (MSR '26), April 13–14, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3793302.3793592

1 Introduction

AI-agents are gaining an important place in software development [21]. They are part of software teams and can fully participate in the development of new features, fixing bugs, and improving existing code [21]. These tasks arise from natural language prompts in which a developer initiates a task for an AI-agent to complete.

Toward a vision of AI-agents being completely autonomous and capable of participating as productive and efficient contributors, we should achieve a deep understanding of the limitations and the failure modes of AI-agents. From the AIDev dataset, we observe that 46.41% of the fixes created or co-authored by Copilot, Devin, Cursor, or Claude end up being rejected. Understanding the reasons behind these rejections is critical to better guide practitioners on how to use agents and inform future studies on how to help developers better integrate these AI-agents. Besides, for certain LLMs, the generated code that ends up being rejected wastes AI-agent tokens or limited premium queries. Another motivation for the need to understand the failing modes of AI-agents is the trustworthiness of agents; lack of trust can affect the adoption of AI-agents [16, 19].

In this paper, we study the reasons why developers reject AI-agent fixes and quantify the effort of review caused by each reason, seen as a wasted effort. The effort we measure through the code churn and the number of comments. The higher the code churn, the more effort one has to investigate on the generated code before reaching the conclusion of rejecting the fix. We focus on fixes for several reasons: (1) fixing bugs is about restoring the correct behavior of software; hence, their rejection is more likely to be grounded in technical issues rather than users, developers', or product-based preferences; (2) the tasks are more likely to be scoped around a given fine-grained task of fixing a failure; and (3) AI-agent pull requests are second in terms of dominance, yet first in terms of rejection rate according to the AI-Dev dataset.

The closest work to our paper is a recent study [26] that analyzed the rejection reasons for Claude Agentic-PRs across heterogeneous task types. Our work differs fundamentally in three key ways. (1) Their analysis is limited to a single agent (Claude), while we look at four agents. (2) Most importantly, their analysis aggregate pull requests with different intents (e.g., refactoring, new features, documentation, etc.), while we exclusively focus on bug

fixing agentic-PRs. Combining all types of changes might hide relevant reasons specific to fixing issues and under-represent specific reasons that are more relevant for fixing bugs and less relevant for other types of task. Focusing on only bugs will guide future studies on providing customized solutions for fixing bugs, which is aligned with the whole software engineering literature that has distinct sub-fields for fixing bugs, creating new features, writing high quality documentation, etc.. For example, a previous work [24] specifically looked at the rejection of pull requests for refactoring. (3) The prior work on Claude manually studied only 92 rejected PRs, while we studied a larger and representative sample of 306 rejected fixes. Replicating the findings on a larger dataset on itself without the other differences is necessary for PR acceptance [17].

In this paper, we perform a qualitative analysis on a representative sample of 306 rejected AI-agentic fixes to identify the reasons for the rejection and a quantitative analysis to better understand these reasons. We addresses the following research questions:

RQ1. Why developers reject AI-agents generated fixes? We qualitatively investigate the reasons for rejecting AI-agent fixes. We find that rejected issues are due to incorrect requirements, incorrect implementation (e.g., wrong approach), technical issues related to the fix (e.g., failing CI), or the fix itself is low priority.

RQ2. How different are rejection reasons in terms of code changes? We quantitatively investigate the wasted effort in terms of code churn and exchanged comments, associated with the rejection of AI-agent fixes. We find that those fixes incur a substantial amount of code churn ranging from a median of 81 to 293.

Take-home message: Our findings suggest that developers should better guide the agents on how to approach the issue, identifying the approaches that can be considered invalid, discussing how to validate the fixes so that they pass tests and do not break CI pipelines, and minimizing the use of AI-agents for low priority fixes. Our findings suggest further studies on how to design and phrase prompts for fixing issues.

Our replication package is made publicly available [15].

2 Data Collection

The study uses the AIDev dataset [21], which includes pull requests generated by AI coding agents across various open-source GitHub repositories. We leverage the AIDev dataset with more than 100 stars, comprising 33,596 curated pull requests distributed over 2,807 projects. As we wish to better understand the reasons for rejecting agent-generated fixes, we consider pull requests that are (i) closed and not merged, (ii) created or co-authored by Copilot, Devin, Cursor, or Claude, and (iii) propose agent code fixes. We exclude Codex pull requests since they are just labeled with "codex", and we cannot systematically identify their contribution to the fixes. This results in 1,497 rejected AI-agentic fixes, from which we select a representative random sample (confidence level = 95% and margin of error = 5%) of 306 pull requests to study. The AIDev dataset features a total of 3,225 fix pull requests contributed by the four agents, out of which 1,497 were rejected fixes, forming **46.41%** of the fixes.

3 Results

RQ1. Why developers reject AI-agents generated fixes?

Motivation: This research question aims to deepen our understanding of the reasons developers reject agent-suggested code fixes. Such insights can help practitioners leverage AI-based coding agents more effectively, clarifying when such agents fall short and guiding future research on how to improve the integration of AI-agents as reliable teammates.

Approach: We adopt a qualitative analysis technique to identify the main reasons for rejecting AI-agent code fixes. Two authors independently label the 306 pull requests, by carefully reading the pull request discussion threads (i.e., commensts, CI results, external artifacts) to extract evidence explaining the rejection of AI-agent fixes. Second, based on the derived context, each author assigns an initial label that captures the underlying reason for rejection. Through a discussion, we come up with a final set of labels, which is used to label all the data. With the final labels, we assess the inter-rater reliability between the two raters using Cohen's κ, achieving a value of 0.605, which indicates substantial agreement. To resolve the disagreement cases, the third author conducts an analysis on the 93 conflicting cases reviewing them to reach a consensus. Then, all the three authors together through a discussion regroup the obtained labels into high level ones.

Results: We identify four high-level themes underlying the rejection of AI-agent fixes, which are further decomposed into 14 sub-themes. Table 1 summarizes these themes along with their definitions and illustrative examples, and we discuss them in detail in the following findings. Note that we were unable to label 151 agentic-PRs since they have no clear reason for rejection.

Relevance-related issues are the most common reason for rejecting AI-agent fixes. This category captures agentic pull requests whose proposed fixes become less relevant over time. This high-level theme is further divided into five sub-themes. The most prevalent sub-theme belongs to pull requests that remain inactive for long periods [7], accounting for 17.3% of the cases. We also observe that superseded pull requests [5] form the second emeging sub-theme with 5.9%, typically occurring when other pull requests address the same issue, propose a better alternative, or replace the original fix. The remaining sub-themes are less frequent and include low-priority pull requests (three cases) [9], one case related to architectural changes [6], and one for testing purposes [4].

These results highlight the need for an approach that prioritizes issues to fix in order to avoid creating duplicates or pull requests that remain inactive with no further activity. These can be seen as waste in terms of agentic-AI resource consumptions (e.g., compute, tokens for token-based models and premium queries for agents like Copilot). They are also similar to the abandonned changes that are well known in the literature as an important waste [20, 23, 25].

AI-agent fixes can be rejected when they are not properly implemented, forming the second major rejection theme. As shown in Table 1, implementation-related issues refer to changes that follow an incorrect approach and are divided into five sub-themes. The most common sub-themes involve pull requests closed due to an incorrect fix [8] or an inappropriate solution approach [14], accounting for 5.6% and 2.6% of the qualitatively analyzed cases, respectively. Similarly, AI-agent fixes are rejected when requirements are ambiguous [13] or not adequately addressed [1], or when changes are made in the wrong repository [11].

Table 1: Taxonomy of rejection reasons for AI-Agent pull requests.

Category	Sub-category	Definition	Example	# (%)
Relevance of the fix	Inactivity	The pull request was automatically closed after a predefined period triggered by the absence of interaction.	*"Closing due to inactivity for more than 7 days"*	53 (17.3%)
	Superseded	PRs that were closed because they were replaced by other PRs or a better alternative.	*"Closed in favor of #1356"*	18 (5.9%)
	Low priority	The PR was closed because the issue is minor or non-critical.	*"This seems like a low prio issue [...]"*	3 (1.0%)
	Architecture	Closed because the architecture or framework changed significantly.	*"[...] major architectural change [...]"*	1 (0.3%)
	Test PR	PRs that were created only for testing purposes.	*"test PR"*	1 (0.3%)
Implementation issues	Incorrect fix	PRs closed due to functionally incorrect or incomplete changes.	*"this PR is not doing what it says [...]"*	17 (5.6%)
	Wrong approach	PRs closed because the issue was resolved through an alternative implementation.	*"@copilot – went a different way [...]"*	8 (2.6%)
	Ambiguity	Changes based on an incorrect understanding of requirements.	*"I think it's confused because [...]"*	2 (0.7%)
	Insufficient	The PR partially fixes the issue but leaves key functionality incomplete.	*"[...] the detail property is not provided"*	2 (0.7%)
	Wrong repo	Submitted to the wrong repository.	*"Plz PR to eliza-plugins org [...]"*	2 (0.7%)
Provider	Agent failure	AI agent failed, became unreachable, or produced erroneous output.	*"Devin is currently unreachable [...]"*	23 (7.5%)
	Rate Limit	Could not be executed due to API or resource restrictions.	*"[...] hit a rate limit [...]"*	3 (1.0%)
Technical issues	CI failure	PRs closed because CI pipelines or automated tests failed.	*"build still fails [...]"*	21 (6.9%)
	Breaking change	Introduced unintended side effects or broke existing functionality.	*"This implementation introduces a breaking change."*	1 (0.3%)
Others	Others	N/A	N/A	151 (49.3%)

These findings suggest that clearer guidance on preferred solutions—and explicit constraints on undesirable approaches—can better guide AI agents. This guidance can be provided through project-specific instruction files [18], or explicitly included in prompts or GitHub issues related to the PRs.

Technical-related issues constitute another category of AI-agent pull request rejections, comprising two sub-themes: *CI failures* and *breaking changes*, accounting for 21 and 1 PRs, respectively. In CI-related rejections, we observe cases where tests fail or infrastructure issues prevent the agent from successfully landing the fix. A good example is shown in Table 1, where the CI pipeline repeatedly failed during the build process [3]. In contrast, only one case involved an implementation that introduced a breaking change [12].

Consequently, this category indicates that AI agents often lack sufficient contextual information on how to write, configure, and execute tests that pass successfully. Providing agents with clearer testing guidelines and CI-related instructions may help reduce such rejections and improve task execution.

Provider-related issues are also a common reason for rejecting AI-agent pull requests. Agents may fail to complete tasks due to provider-side problems, such as becoming unreachable, producing incorrect responses, or failing during execution; this sub-theme accounts for 23 pull requests. Note that we also consider pull requests with no commits in this category. For example, the Devin agent encountered issues when responding to a developer's query [10], as reflected in the message: *"Devin is currently unreachable—the session may have died. Let's just do that and then close this one."*. Second, provider-imposed rate limits can prevent agents from continuing their work, a sub-theme observed in three cases. For instance, Copilot reached the maximum number of allowed requests [2], as noted in the message: *"Sorry, you've hit a rate limit that restricts the number of Copilot model requests [...]."*. About the rate limit of Copilot, any request to the agent can be seen as a new request, which quickly sums up and reach the limit. These findings suggest that provider-specific constraints can directly impact the success of AI-agent pull requests.

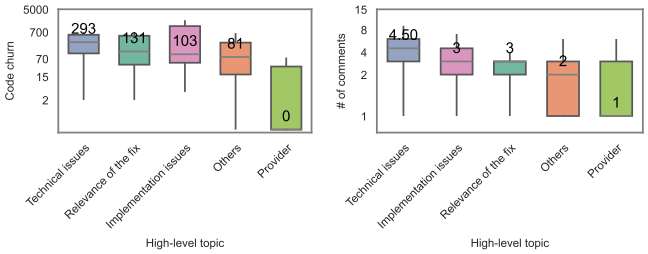
Summary

We observe 14 reasons for rejecting Agentic-PRs for fixing bugs. They are related to the implementation that can be incorrect, face issues on the CI pipeline, the agent failed to perform the task, or the fix is rejected for irrelevance.

RQ2. How different are rejection reasons in terms of code changes?

Motivation: The goal of this research question is to quantify the number of changes made by agents that end up being rejected for each of our five identified reasons from RQ1. The greater the number of changes, the more effort is required to review the change, and the higher AI-agents resources are wasted.

Approach: To answer this RQ, we collect the code churn and the number of commits for each pull request using the AI-Dev dataset and compare the five categories with each other.



(a) Code churn **(b) Comments**

Figure 1: The code churn and discussion comments across rejected PR topics.

Results: Rejected PRs have a large code churn with a median that ranges from 81 to 293, as shown in Figure 1a. The Provider category has a median of zero, since we consider changes with zero commits in that category (the agent was unable to generate code). We observe that the category with the highest code churn is related to the pull requests that face technical issues. That can be explained by the fact that the larger the change, the more likely it will lead to build failures. That can suggest further studies to look into these failing tests. One assumption is that the AI-agents with a large amount of changes might need to adjust the tests as well. The agents should also have guidance on how to handle these failures, by potentially providing tests that should be run for a given fix. The implementation issue can also introduce an important waste as developers need to review a large amount of changes (a median of 103 lines) to find out that the generated fix is inadequate.

Although we do not observe a large number of comments, the rejected pull requests still received comments. In fact, the median number of comments is between one and 4.5, as depicted in Figure 1b. However, this number is not different from that of the merged fixes. The merged fixes have a median of two comments (Q1 and Q3 of one and 4). As such, and since the number of rejected pull requests constitutes almost half of the data (46.41% as discussed in the Data Collection Section), almost half of the comments are wasted. Even for tasks that are irrelevant, they have a median of three comments before rejecting the pull request.

Summary

Rejected Agentic-PRs have a median code churn that range between 81 and 293. The higher code churn is associated with the technical issues (e.g., CI-failure), followed by the Agentic-PRs rejected for their relevance then implementation issues. Rejected fixes involve a median of one to 4.50 comments similarly to the merged ones (a median of of two).

4 Implications

This section discusses some implications of our findings.

Implication 1: We suggest that developers provide guidance on how to perform the fix. We observe that a good amount of rejected fixes are due to an incorrect way of fixing issues. As such, we suggest that developers provide guidance on how to perform the fix or provide guidance on what approaches are not acceptable in the agent instruction file (e.g., .github/copilot-instructions.md).

Implication 2: We suggest developers provide agents with instructions on how to validate a change. A good amount of changes are rejected because the issue is not fixed, fixes fail tests or CI pipelines. Guiding the agent on how to run tests and how to validate if the issue is fixed or would not avoid agents generate inadequate fixes. That is in addition to further studies on understanding the CI failures and how to instruct agents to avoid them.

Implication 3: Better task management is required to minimize using AI-agents for low priority tasks that end up being closed. These tasks despite their low priority, agents generate a large amount of changes for them which require human reviews and testing such that they do not break the code. Irrelevant and low priority fixes should be avoided at the first place to avoid generating irrelevant code that would also involve human reviews. Although a large body of prior research [22, 27] focused on duplicate patches and pull requests, these studies need to be replicated to avoid agents creating duplicate fixes from the prompt itself. As such, studies are required to evaluate the priority of tasks all the way from their requirements before assigning them to AI-agents.

Implication 4: We suggest researchers develop approaches (e.g., templates) that better guide developers prompt the AI-agents for fixing issues; Our study enumerates a set of reasons for rejecting the AI-agents fixes, which can motivate future studies to develop quality metrics that assess the quality of a prompt such that the AI-agent implements a successful fix. For example, does the issue enumerate its relevant CI tests?

5 Conclusion and Future Work

46.41% of the fixes created by Copilot, Devin, Cursor, and Claude are rejected. Understanding the rejection of these fixes is critical to determine AI-agents fall short, such that developers can better integrate agents in their pipelines and researchers can provide solutions to guide developers on better use of agents to fix issues autonomously (aka., **autonomous AI-agents**).

Toward this goal, we conduct a qualitative study on the reasons for rejecting fixes. We find four high-level reasons that encourage developers to provide guidance on how to perform the fix, guidance on how to validate if the change fixed the issue, and guidance on how to prioritize relevant issues (not low priority ones). The categories obtained are likely to involve a good amount of wasted resources, coming from the amount of generated code (AI-agent resources) and human efforts to review the generated fixes. Almost half of the comments are around rejected fixes.

In the future, we plan to design approaches that can better guide developers in using AI-agents for successfully fixing issues. For instance, we plan to provide and evaluate a prompt template, evaluate the contribution of instruction files on improving the AI-agents, design approaches that recommend tests relevant to an issue, and that can be used as a validation point for generated fixes.

References

[1] 2025. Pull Request #10408. https://github.com/vercel/turborepo/pull/10408. Accessed on 2025-12-27.

[2] 2025. Pull Request #11352. https://github.com/taiga-family/taiga-ui/pull/11352. Accessed on 2025-12-27.

[3] 2025. Pull Request #1199. https://github.com/christianhelle/apiclientcodegen/pull/1199. Accessed on 2025-12-27.

[4] 2025. Pull Request #12466. https://github.com/Azure/Azure-Sentinel/pull/12466. Accessed on 2025-12-27.

[5] 2025. Pull Request #1353. https://github.com/neondatabase/autoscaling/pull/1353. Accessed on 2025-12-27.

[6] 2025. Pull Request #149. https://github.com/ruvnet/claude-flow/pull/149. Accessed on 2025-12-27.

[7] 2025. Pull Request #1554. https://github.com/567-labs/instructor/pull/1554. Accessed on 2025-12-27.

[8] 2025. Pull Request #219. https://github.com/syncfusion/maui-toolkit/pull/219. Accessed on 2025-12-27.

[9] 2025. Pull Request #305. https://github.com/bespokelabsai/curator/pull/305. Accessed on 2025-12-27.

[10] 2025. Pull Request #3113. https://github.com/crewAIInc/crewAI/pull/3113. Accessed on 2025-12-27.

[11] 2025. Pull Request #4354. https://github.com/owncast/owncast/pull/4354. Accessed on 2025-12-27.

[12] 2025. Pull Request #50357. https://github.com/Azure/azure-sdk-for-net/pull/50357. Accessed on 2025-12-27.

[13] 2025. Pull Request #61902. https://github.com/microsoft/TypeScript/pull/61902. Accessed on 2025-12-27.

[14] 2025. Pull Request #75. https://github.com/rqlite/sql/pull/75. Accessed on 2025-12-27.

[15] 2025. Replication Package for: Understanding the Rejection of Fixes Generated by Agentic Pull Requests - Insights from the AIDev Dataset. doi:10.6084/m9.figshare.30964363

[16] Agathe Balayn, Mireia Yurrita, Fanny Rancourt, Fabio Casati, and Ujwal Gadiraju. 2024. An Empirical Exploration of Trust Dynamics in LLM Supply Chains. *arXiv preprint arXiv:2405.16310* (2024).

[17] Di Chen, Kathyrn T Stolee, and Tim Menzies. 2019. Replication can improve prior results: A github study of pull request acceptance. In *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*. IEEE, 179–190.

[18] GitHub. 2024. *Adding custom instructions for GitHub Copilot.* https://docs.github.com/en/copilot/how-tos/configure-custom-instructions/add-repository-instructions GitHub Docs.

[19] Dipin Khati. 2025. Trustworthiness of Large Language Models for Code. In *Proceedings of the 2025 IEEE/ACM 47th International Conference on Software Engineering - Companion (ICSE Companion)*. IEEE, Lisbon, Portugal.

[20] Sayedhassan Khatoonabadi, Diego Elias Costa, Rabe Abdalkareem, and Emad Shihab. 2023. On Wasted Contributions: Understanding the Dynamics of Contributor-Abandoned Pull Requests–A Mixed-Methods Study of 10 Large Open-Source Projects. *ACM Trans. Softw. Eng. Methodol.* 32, 1, Article 15 (Feb. 2023), 39 pages. doi:10.1145/3530785

[21] Hao Li, Haoxiang Zhang, and Ahmed E. Hassan. 2025. The Rise of AI Teammates in Software Engineering (SE) 3.0: How Autonomous Coding Agents Are Reshaping Software Engineering. arXiv:2507.15003 [cs.SE] https://arxiv.org/abs/2507.15003

[22] Zhixing Li, Gang Yin, Yue Yu, Tao Wang, and Huaimin Wang. 2017. Detecting Duplicate Pull-requests in GitHub. In *Proceedings of the 9th Asia-Pacific Symposium on Internetware* (Shanghai, China) *(Internetware '17)*. Association for Computing Machinery, New York, NY, USA, Article 20, 6 pages. doi:10.1145/3131704.3131725

[23] Zhixing Li, Yue Yu, Tao Wang, Gang Yin, ShanShan Li, and Huaimin Wang. 2022. Are You Still Working on This? An Empirical Study on Pull Request Abandonment. *IEEE Transactions on Software Engineering* 48, 6 (2022), 2173–2188. doi:10.1109/TSE.2021.3053403

[24] Jevgenija Pantiuchina, Bin Lin, Fiorella Zampetti, Massimiliano Di Penta, Michele Lanza, and Gabriele Bavota. 2021. Why do developers reject refactorings in open-source projects? *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 2 (2021), 1–23.

[25] Qingye Wang, Xin Xia, David Lo, and Shanping Li. 2019. Why is my code change abandoned? *Information and Software Technology* 110 (2019), 108–120. doi:10.1016/j.infsof.2019.02.007

[26] Miku Watanabe, Hao Li, Yutaro Kashiwa, Brittany Reid, Hajimu Iida, and Ahmed E Hassan. 2025. On the use of agentic coding: An empirical study of pull requests on github. *arXiv preprint arXiv:2509.14745* (2025).

[27] Yue Yu, Zhixing Li, Gang Yin, Tao Wang, and Huaimin Wang. 2018. A dataset of duplicate pull-requests in github. In *Proceedings of the 15th International Conference on Mining Software Repositories* (Gothenburg, Sweden) *(MSR '18)*. Association for Computing Machinery, New York, NY, USA, 22–25. doi:10.1145/3196398.3196455