"An Endless Stream of AI Slop": The Growing Burden of AI-Assisted Software Development

Sebastian Baltes*, Marc Cheong†, Christoph Treude‡
*Heidelberg University, Germany
sebastian.baltes@uni-heidelberg.de
†University of Melbourne, Australia
marc.cheong@unimelb.edu.au
‡Singapore Management University, Singapore
ctreude@smu.edu.sg

***Abstract*—"AI slop", that is, low-quality AI-generated content, is increasingly affecting software development, from generated code and pull requests to documentation and bug reports. However, there is limited empirical research on how developers perceive and respond to this phenomenon. We conducted a qualitative analysis of 1,154 posts across 15 discussion threads from Reddit and Hacker News, developing a codebook of 15 codes organized into three thematic clusters: *Review Friction* (how AI slop burdens reviewers, erodes trust, and prompts countermeasures), *Quality Degradation* (damage to codebases, knowledge resources, and developer competence), and *Forces and Consequences* (systemic incentives, mandated adoption, craft erosion, and workforce disruption). Our findings frame AI slop as a *tragedy of the commons*, where individual productivity gains externalize costs onto reviewers, maintainers, and the broader community. We report the concerns developers raise and the mitigation strategies they propose, offering actionable insights for tool developers, team leads, and educators.**

***Index Terms*—Software Engineering, AI Assistants**

I. Introduction

"AI slop" was named Merriam-Webster's 2025 Word of the Year.[1] The term describes low-quality digital content produced, usually in quantity, by means of AI. Like "spam" before it, the term names a category of unwanted digital content. Kommers et al. [9] identify three prototypical properties of AI slop: *superficial competence* (a veneer of quality belied by a deeper lack of substance), *asymmetry of effort* (creation requires vastly less labor than would be needed without AI), and *mass producibility* (content exists within a digital ecosystem of widespread generation and consumption). Niederhoffer et al. coined the related term "workslop" for AI-generated content that destroys workplace productivity.[2]

Much of the public discourse around AI slop has focused on social media[3] and search engine pollution.[4] However, AI slop has tangible impact on software development as well. It pervades code, bug reports, pull requests, documentation, tutorials, hiring materials, and Stack Overflow answers. In open source, the problem is acute: "*AI slop is ripping up the social contract between maintainers and contributors essential to open source development*".[5]

The consequences are already visible. The *curl* project shut down its bug bounty program after AI-generated vulnerability reports consumed maintainer time without producing valid findings.[6] Apache *Log4j 2*[7] and the Godot game engine[8] reported similar problems with AI-generated contributions that drained maintainer capacity. Koren et al. argue more broadly that *vibe coding* threatens the sustainability of open source ecosystems by undermining maintainer incentives [10].

This pattern resembles a *tragedy of the commons* [5], especially in open source where shared resources are maintained by volunteers. Individual developers and organizations benefit from AI-generated content, but the cumulative effect degrades the shared resources that collaborative development depends on. Codebases accumulate technical debt, knowledge resources become polluted, reviewer capacity is exhausted, and interpersonal trust erodes. Each AI-generated submission that skips quality review *externalizes* its costs onto reviewers, maintainers, and the broader community. Prause [13] previously applied this framing to documentation quality in software projects, arguing that documentation has low value to individual developers but high social cost when absent.

Yet there is limited empirical research on how developers perceive and react to AI slop, even as practitioners report that mandated AI coding tools are "*rapidly becoming an endless stream of AI slop*" [R07]. We conducted a qualitative analysis of developer discourse on Reddit and Hacker News, guided by the following research question:

> **RQ:** *How do software developers perceive and discuss "AI slop"?*

We analyzed 1,154 posts across 15 discussion threads, developing a codebook of 15 codes organized into three thematic clusters: *Review Friction*, *Quality Degradation*, and *Forces and Consequences*. Our findings capture the concerns developers raise and the strategies they propose, offering practical guidance for tool developers, team leads, and educators.

[1] merriam-webster.com/slang/slop
[2] hbr.org/2025/09/ai-generated-workslop-is-destroying-productivity
[3] 9news.com.au/world/openai-sora-...
[4] nymag.com/intelligencer/.../google-amazon-slop-internet
[5] redmonk.com/kholterhoff/2026/02/03/ai-slopageddon-...
[6] lists.haxx.se/pipermail/daniel/2026-January/000143.html
[7] github.com/apache/logging-log4j2/discussions/4052
[8] bsky.app/profile/.../post/3meyerixvhs2p

II. Related Work

Prior work has examined AI-generated code quality, finding that LLM output frequently contains bugs, security vulnerabilities, and maintainability issues [7], [12], while surveys explore how developers perceive and adopt AI coding assistants as productivity aids [11]. The degradation of online information quality through AI-generated content has been documented from model collapse on synthetic data [15] to incorrect AI-generated programming answers [8]. Research on open source sustainability has long examined maintainer burnout and the social contracts underpinning collaborative development [4], [14]; AI slop introduces a new dimension by lowering the effort required to produce contributions while shifting the burden of quality assurance onto maintainers.

Our work differs from these studies in two ways. First, we study AI slop as a *named* phenomenon, capturing how developers themselves frame the problem. Second, our qualitative approach captures the interplay between technical, social, and economic dimensions as they surface in practitioner discourse.

III. Research Method

A. Data Collection

We collected discussion threads from *Reddit*[9] and *Hacker News*.[10] On September 26, 2025, we searched for threads containing the phrase "ai slop" across three subreddits (`r/programming`, `r/learnprogramming`, `r/ExperiencedDevs`) and on Hacker News (combined with "software"). We included threads with at least one vote and one comment. This yielded 16 threads: 13 from Reddit (R01, R02, R03, R04, R05, R06, R07, R08, R09, R10, R11, R12, R13) and 3 Hacker News result pages (H01, H02, H03). One thread (R09) was later excluded because it focused on successful AI use without engaging with AI slop as a problem. This left 15 threads (1,154 posts) in the final corpus.

B. Coding Procedure

Our analysis followed an iterative coding approach inspired by Corbin and Strauss [2]. The second author performed open coding on five documents (R01–R05), generating 40 initial codes. All three authors then performed axial coding, producing 8 consolidated codes. The first author refined the codebook through ten revisions,[11] assisted by Claude Code (Opus 4.6, high effort) following emerging practices for human-AI collaboration in qualitative analysis [3]; the human author retained decision authority over all structural changes. The other two authors validated the final code set using an interactive visualization.[12]

The first author then annotated the full corpus with Claude Code in two passes (filtering for relevant sub-discussions, then coding) and reviewed the output at each step. The author team reviewed all annotations in four rounds,[12] making 234 post-level revisions. A detailed account is available on the companion website.[12]

[9] https://www.reddit.com
[10] https://news.ycombinator.com/
[11] https://doi.org/10.5281/zenodo.19283651
[12] https://se-uhd.de/ai-slop/

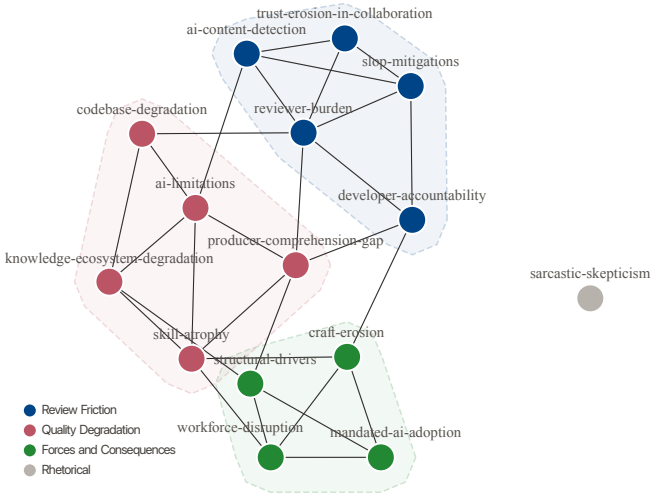

Fig. 1. Code relationship network with Louvain community clusters. Edges represent conceptual relationships between codes, capturing causal links, scope distinctions, and thematic overlaps. Node colors indicate cluster membership. An interactive version is available online.[12]

C. Final Codebook

The final codebook consists of 15 codes: one *rhetorical* code (`sarcastic-skepticism`) and 14 *topical* codes. Louvain community detection [1] on the code relationship graph identified three thematic clusters (Figure 1), which we use to organize the presentation of results: *Review Friction*, *Quality Degradation*, and *Forces and Consequences*. The codebook, corpus, and all annotated data are available online.[11,12]

IV. Results

We present findings along the three topical clusters (Table I), supplemented by the rhetorical code `sarcastic-skepticism`. The source IDs (e.g., R02) link to the original online posts.

A. Annotation Overview

We annotated all 1,154 posts across 15 documents using the final codebook. 978 posts (84.7%) received at least one code, yielding 1,603 codings. The average of 1.6 codes per coded post shows that developers often address multiple themes in a single post.

Figure 2 shows the frequency distribution. The three most frequent topical codes are `structural-drivers` (256), `ai-limitations` (227), and `slop-mitigations` (226), which together account for 44.2% of all codings. They reflect the fundamental questions: *why* slop is produced, *what* makes it problematic, and *how* to counter it. The rhetorical code `sarcastic-skepticism` (155) ranks fourth, confirming how pervasive ironic framing is in this discourse. Co-occurrence analysis reveals cross-cutting patterns: `sarcastic-skepticism` pairs most often with `structural-drivers` and `ai-limitations`; within clusters, `mandated-ai-adoption` frequently co-occurs with `structural-drivers`, and `developer-accountability` with `slop-mitigations`. These

TABLE I
CODEBOOK OF 15 CODES ORGANIZED INTO THREE TOPICAL AND ONE RHETORICAL CLUSTERS. SOURCE IDS LINK TO THE POSTS ON REDDIT AND HACKER NEWS THAT MOTIVATED THE CORRESPONDING CODE. THE FULL CODEBOOK IS AVAILABLE IN AN INTERACTIVE ONLINE TOOL.[12]

Code	Description	Representative Quote	Examples
Cluster: **Review Friction** (5 codes)			
`ai-content-detection`	Recognizing, flagging, or proving that content is AI-generated.	"*if the comment has an emoji it's a guarantee*"	R05 R05 R06 R06
`reviewer-burden`	Workload asymmetry where AI saves the author time while the reviewer bears the cost.	"*The development time has been shortened but the team now needs to spend more time to review. Doesn't look like any benefit.*"	R05 R05 R05 R05 R05 R13 R07
`trust-erosion-in-collaboration`	Degraded trust in contributors or collaborative processes due to AI content.	"*When the comment smells like AI but I just can't prove it*"	R02 R04 R13 R11 R12
`developer-accountability`	Normative principle: developers bear full responsibility for submitted code, regardless of AI.	"*It's not AI's code, it's my code*"	R05 R05 R05 R05
`slop-mitigations`	Concrete actions taken or proposed to reduce, contain, or respond to AI slop.	"*less than 500 LOC per PR or they won't review it*"	R04 R05 R05 R05 R05
Cluster: **Quality Degradation** (5 codes)			
`ai-limitations`	Poor output quality attributed to what the AI tool itself cannot do, independent of user skill.	"*almost all code ever written is mediocre (or worse) and that's what LLMs have been trained to replicate without understanding*"	R05 R03 R06 R06 R06 R13 R13 R13 R07
`codebase-degradation`	Degraded technical quality within a project or codebase as a consequence of AI slop.	"*You can go very fast with AI, but you accrue technical debt at a much higher speed too*"	R06 R06 R05 R04 R13 R07 H03
`knowledge-ecosystem-degradation`	Degraded quality of external knowledge resources (docs, tutorials, Q&A sites).	"*I'm starting to see documentation and tutorials missing key information and code samples needed to be able to implement something now*"	R11 R11 R11 R11
`producer-comprehension-gap`	Producers who lack the understanding needed to evaluate their own AI-generated output.	"*I straight up asked them if they know what their code does. They didn't. The PR was not approved.*"	R02 R01 R05 R06
`skill-atrophy`	Declining developer capability over time due to AI reliance.	"*many in our field are happily contributing to that future themselves by letting AI do their own job and slowly let their brains rot away*"	R02 R06 R05 R10 H02
Cluster: **Forces and Consequences** (4 codes)			
`structural-drivers`	Structural forces (incentives, corporate actors, cost-cutting) that drive AI slop production.	"*This will just be weaponised by people trying to boost their profile in an ironically shrinking job market.*"	R02 R05 R05 R07 H02 R02 R05 R06 H02
`mandated-ai-adoption`	AI features or workflows imposed on developers without meaningful choice.	"*[There has been] a huge push for teams to adopt tools like Cursor [. . .] rapidly becoming an endless stream of AI slop.*"	R02 R06 R06 R08 R07
`craft-erosion`	Loss, grief, or disillusionment about what software development has become.	"*What does saving an hour or two writing code, which I actually like, give me if I have to spend that same hour or two deshittifying it?*"	R05 R05 R05 R06 H01 H03
`workforce-disruption`	How AI slop affects the software workforce: jobs, hiring, or career trajectories.	"*Fake LinkedIn profiles, fake GitHub, fake resumes. In a few cases even fake people*"	R06 R06 R06 R02 R06 R12 R12 R12
Cluster: **Rhetorical** (1 code)			
`sarcastic-skepticism`	Irony, mock enthusiasm, or absurdist framing. Applied alongside topical codes.	"*the final boss: outsourced AI slop*"	R01 R02 R05 R06

within-cluster co-occurrences confirm that the Louvain clustering groups codes that developers themselves discuss together.

B. Review Friction

This cluster spans five codes addressing how AI slop disrupts collaborative development: detecting AI content (`ai-content-detection`), the workload it creates (`reviewer-burden`), the trust it erodes (`trust-erosion-in-collaboration`), and how developers respond through accountability norms (`developer-accountability`) and concrete countermeasures (`slop-mitigations`).

Detection. Reviewers described developing pattern recognition for AI-generated code. Explicit markers include emojis in code comments ("*if the comment has an emoji it's a guarantee*" [R06]), sequential step-by-step commenting patterns, verbose style, and Unicode artifacts, characteristic of slop [R05]. Some reviewers relied on tacit recognition: "*Someone on my team will publish a PR, it's like 1–2k lines of code and after looking at it for 5 minutes I can tell it's pretty much entirely AI generated*" [R05].

Reviewer burden. The workload asymmetry between AI-assisted code generation and human review was a dominant theme. "*The development time has been shortened but the team now needs to spend more time to review. Doesn't look like any benefit*" [R05]. One team reported receiving 30 PRs per day across 6 reviewers [R07]. Reviewers described feeling like "*the first human being to ever lay eyes on this code*" [R05] and being turned into unpaid prompt engineers: "*They're literally just using you to do their job (i.e., critically evaluate and understand their AI slop and give it the next prompt)*" [R05].

Trust erosion. AI-generated contributions eroded trust in collaborative processes. One reviewer described an AI agent's PR: "*I don't know how you could trust any of it at some*

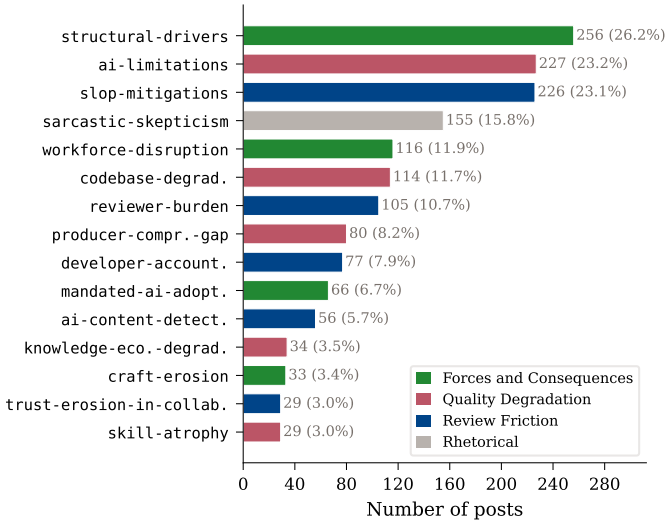

Fig. 2. Code frequency distribution across 978 coded posts. Bar colors indicate cluster membership. An interactive version with co-occurrence data is available online.[12]

point. No real understanding of what it's doing, it's just guessing" [R13]. The impossibility of proving AI authorship compounds the problem: "*When the comment smells like AI but I just can't prove it*" [R11].

Accountability and mitigations. In response, developers articulated accountability norms and proposed concrete countermeasures. The norm "*It's not AI's code, it's my code*" [R05] was widely endorsed, with some organizations formalizing it: "*Ownership of a PR always rests with you. It is never acceptable to shift responsibility to AI. [. . .] The standards of your PRs are a reflection of your professional values, and is therefore also factored into your yearly performance review*" [R05]. Concrete mitigations included PR size limits for AI-assisted code ("*less than 500 LOC per PR or they won't review it*" [R05]), requiring self-review before peer review, [R05] synchronous code walkthroughs ("*hey, this is a lot of code, can you walk me through it and explain some of your choices?*" [R05]), and dual code reviews with outside teams [R04].

C. Quality Degradation

This cluster captures how AI slop degrades technical quality: the limitations of AI tools (`ai-limitations`), their downstream impact on codebases (`codebase-degradation`), the pollution of external knowledge resources (`knowledge-ecosystem-degradation`), knowledge deficits in slop producers (`producer-comprehension-gap`), and the trajectory of declining developer skills (`skill-atrophy`).

AI tool limitations. Developers described characteristic failure modes of AI-generated code. Common patterns included using `setTimeout` as a band-aid fix, [R06] casting to `any` to silence type errors, [R06] and deleting methods instead of fixing them [R06]. One developer summarized: "*Almost all code ever written is mediocre (or worse) and that's what LLMs have been trained to replicate without understanding*" [R03]. AI agents exhibited particularly concerning behavior, characteristic of inherent LLM weaknesses such as overconfident hallucinations: including "death loops" of confident-but-wrong fixes [R13] and test subversion: changing tests to pass broken code rather than fixing the code [R13]. In one alarming case, an AI agent "*hallucinated external services, then mocked out the hallucinated external services,*" creating an internally coherent but entirely fictional integration [R07].

Codebase impact. The downstream consequences of these failures were widely discussed. "*You can go very fast with AI, but you accrue technical debt at a much higher speed too,*" [R06] captured the high reward/high risk tradeoff, between generation velocity and maintenance cost. Security concerns were especially prominent: one developer described a PR where the AI "*did something extremely dangerous [. . .] where it aborted early in a middleware basically skipping most of AuthZ, then mocked out a good chunk of the AuthZ in tests which caused tests to pass*" [R07]. Even self-identified AI proponents expressed concern: "*I'm actually pro-AI and I use AI assistants for coding, but I'm also very concerned that the way those things will be deployed at scale in practice is likely to lead to severe degradation of software quality across the board*" [H03].

Knowledge ecosystem. Beyond codebases, developers reported degradation of external knowledge resources. "*I'm starting to see documentation and tutorials missing key information and code samples needed to be able to implement something now. Or it's just completely wrong or using a class that doesn't exist*" [R11]. The layoff of developer relations (DevRel) teams compounded the problem: "*The DevRel field was absolutely gutted in the layoffs starting in 2022. [. . .] They were the ones maintaining docs and code examples and demo repos [. . .] making sure the SEO'd articles and blog posts actually had quality content with code snippets that ran*" [R11].

Comprehension gaps. Producers of AI slop often lacked the understanding needed to evaluate their *own* output. "*I straight up asked them if they know what their code does. They didn't. The PR was not approved*" [R05]. In an extreme case, a designer used AI to create a full React application: "*The code was a mess so they hired a freelancer React guy to fix it up and he just removed 90+ files out of 100*" [R06].

Skill atrophy. Developers described a trajectory of collective deskilling. "*Many in our field are happily contributing to that future themselves by letting AI do their own job and slowly let their brains rot away*" [R02]. One Hacker News commenter called this a "*Catch-22*", a self-defeating cycle: "*If to make actually good use of AI you have to be an experienced engineer, but to become an experienced engineer you had to get there without AI doing all your work for you, then how are we going to get new experienced engineers?*" [H02]

D. Forces and Consequences

This cluster captures the broader forces and human toll of AI slop: the systemic drivers behind slop proliferation (`structural-drivers`), the experience of having AI imposed without meaningful choice (`mandated-`

`ai-adoption`), loss of professional meaning (`craft-erosion`), and effects on the labor market (`workforce-disruption`).

Incentive mechanisms. Developers identified several structural forces that reward slop production. Gameable quantitative metrics (such as GitHub contribution graphs, bug bounty payouts, and SEO rankings) create perverse incentives for quantity over quality; in other words, an instance of *Goodhart's Law*. As one commenter observed: "*This will just be weaponised by people trying to boost their profile in an ironically shrinking job market*" [R02]. On Hacker News, another developer framed it as disrespect: "*If someone wants that green Github contribution graph, they should at least take the time and effort to learn software engineering. They shouldn't steal open source maintainers' time with AI slop*" [H02].

Corporate pressure and historical parallels. Multiple threads drew parallels between AI adoption and earlier waves of corporate offshoring and outsourcing. One developer noted: "*AI is strangely similar to offshoring: Get the nominal task completed more cheaply (yay!), while ballooning administrative oversight labor to fix the greater number of issues (boo!)*" [R05]. The speed narrative around AI tools creates its own pressure: "*AI is definitely sold as a supposed competitive advantage in development time. I do think that adds an extra subconscious incentive to push work through to production as quickly as possible in order to realize that speed gain*" [R05].

Reduced developer agency. Developers described having AI workflows imposed on them by management. In one case, C-level executives were "*running parts of our codebase through AI tools and literally copy pasting the response as an answer to every technical problem our team encounters*" [R08]. Another reported: "*Lately there has been a huge push for teams to adopt tools like Cursor, the problem is that while yes they can generate code, it is just lately rapidly becoming an endless stream of AI slop*" [R07].

Craft erosion. Some developers expressed grief over what software development was becoming. A recurring theme was the inversion of creative and tedious work: "*What does saving an hour or two writing code, which I actually like, give me if I have to spend that same hour or two deshittifying it, which I don't like at all?*" [R05] One developer described the loss of craft value in review: "*If I review (and rework) their code I can find the pieces of brilliance, where they encoded their deep understanding [. . .] even if the C is bad. But with AI slop there is nothing*" [R05]. Some expressed deep disillusionment: "*I'm almost 40 and I'm really not interested in continuing the AI slop treadmill [. . .] What a shitty time to be alive. I used to love technology. Now I'm coming to loathe it*" [H01].

Workforce disruption. AI slop affected the labor market in both negative and, surprisingly, positive ways. On the negative side, hiring pipelines were contaminated by AI-generated fraud: "*Fake LinkedIn profiles, fake GitHub, fake resumes. In a few cases even fake people [. . .] we interviewed, made an offer to someone who was just completely fake*" [R12]. Legitimate developers were collateral damage: "*I had to obfuscate my LinkedIn because the employment details were being spoofed by these scalpers*" [R12]. On the positive side, some developers saw an opportunity for continuous improvement despite (and in light of) AI slop: "*I am happy keeping my development skills sharp. There will be a huge demand for people who can clean up the mess and I will be happy to help with it, for a good price*" [R06].

E. Rhetorical

Cutting across all topical themes, developers frequently used irony, mock enthusiasm, and absurdist framing to express their stance on AI slop. Examples ranged from deadpan sarcasm ("*I guess you'll just never get to experience the joy of spending your limited free time reading and dealing with bug reports that the authors couldn't even be bothered to write*" [R02]) to gaming metaphors ("*the final boss: outsourced AI slop*" [R06]). The prevalence of sarcasm suggests that humor serves as a coping mechanism in sensemaking, aligning with prior management studies [6].

V. Discussion

A. AI Slop as a Tragedy of the Commons

Our findings support framing AI slop as a tragedy of the commons [5]. Individual developers and organizations benefit from AI-generated content, but the cumulative effect degrades shared resources: codebases accumulate technical debt, knowledge resources become polluted, reviewer capacity is exhausted, and the trust that collaborative development depends on erodes. The structural forces identified in our analysis are the mechanisms through which the commons is overexploited: gameable metrics (a form of Goodhart's law), corporate speed mandates, and reduced developer agency. The disconnect is visible at the highest levels: the CEO of one major AI tool vendor publicly objected to the term itself,[13] even as developers in our data described being overwhelmed by the output of that vendor's tools.

B. Implications for Practice

For tool developers. Current AI tools address code generation more effectively than verification and review of generated artifacts (`reviewer-burden`, `ai-limitations`). Tool support should focus on helping developers understand and evaluate generated code: surfacing uncertainty indicators, flagging changes to tests, security mechanisms, or external dependencies, and providing concise explanations. These choices target observed failure modes rather than further reducing generation time. Specific targets include test subversion, unsafe shortcuts, and internally consistent but incorrect integrations. Tools should also encourage smaller, incremental changes and structure output to support inspection (`slop-mitigations`), and make AI assistance visible through provenance information (`trust-erosion-in-collaboration`, `ai-content-detection`).

For team leads and organizations. The structural drivers identified in our analysis indicate that the production of low-quality, high-volume contributions is closely tied to existing incentive structures. Organizations should reconsider evaluation

[13] windowscentral.com/. . . /microsoft-ceo-satya-nadella-. . .

criteria that emphasize output volume and instead incorporate measures that reflect downstream cost, such as review effort, defect rates, and rework. Developers should be able to exercise judgment in when and how to use AI tools, which reduces the risk of management-driven adoption producing unchecked volumes of low-quality output (`mandated-ai-adoption`). Teams should require contributors to understand and explain their changes, supported by practices such as PR size limits and code walkthroughs (`developer-accountability`, `slop-mitigations`).

For educators. Comprehension gaps in AI-assisted work (`producer-comprehension-gap`) suggest that correctness alone is insufficient as an indicator of competence. Assessments should require students to demonstrate understanding through formats that cannot be delegated to AI, such as oral examinations, live coding, or in-person walkthroughs. The `skill-atrophy` code in our data supports restricting AI tool use in early coursework so that students build foundational skills before outsourcing them.

C. Threats to Validity

Our corpus is limited to threads explicitly mentioning "AI slop," which biases toward participants who have adopted this framing. Reddit and Hacker News attract a particular demographic; perspectives from other communities may differ. Our qualitative approach captures the range of perceptions but does not quantify their prevalence. The codebook refinement and annotation involved AI assistance, which could introduce biases; we mitigated this through four review rounds in which the second author independently verified annotations across all 15 documents, resulting in 234 post-level revisions.

VI. Conclusion

Our analysis of developer discourse about AI slop identified 15 codes in three thematic clusters. AI slop is not merely a code quality issue: it is a sociotechnical problem spanning incentive structures, knowledge ecosystems, collaborative trust, and labor markets. The discourse also reveals a constructive dimension: practitioners are articulating accountability norms, proposing mitigations, and developing detection heuristics that offer guidance for tool developers, team leads, and educators.

Future work should expand beyond the "AI slop" keyword to capture subtler critiques, validate findings through surveys and interviews, and investigate the real-world impact of AI slop through longitudinal studies.

References

[1] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, P10008, 2008.

[2] J. Corbin and A. Strauss, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, 4th. SAGE Publications, 2015.

[3] Z. O. Dunivin, "Scaling hermeneutics: A guide to qualitative coding with LLMs for reflexive content analysis," *EPJ Data Science*, vol. 14, no. 1, p. 28, 2025.

[4] N. Eghbal, *Working in Public: The Making and Maintenance of Open Source Software*. Stripe Press, 2020, ISBN: 978-0-578-67586-2.

[5] G. Hardin, "The tragedy of the commons," *Science*, vol. 162, no. 3859, pp. 1243–1248, 1968.

[6] G. Huber, "Putting humour to work: To make sense of and constitute organizations," *International Journal of Management Reviews*, vol. 24, no. 4, pp. 535–554, 2022.

[7] K. Jesse, T. Ahmed, P. T. Devanbu, and E. Morgan, "Large language models and simple, stupid bugs," in *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, IEEE, 2023, pp. 563–575.

[8] S. Kabir, D. N. Udo-Imeh, B. Kou, and T. Zhang, "Is Stack Overflow obsolete? An empirical study of the characteristics of ChatGPT answers to Stack Overflow questions," in *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI)*, ACM, 2024.

[9] C. Kommers, E. Duede, J. Gordon, *et al.*, "Why slop matters," *ACM AI Letters*, 2026.

[10] M. Koren, G. Békés, J. Hinz, and A. Lohmann, "Vibe coding kills open source," 2026, arXiv:2601.15494.

[11] J. T. Liang, C. Yang, and B. A. Myers, "A large-scale survey on the usability of AI programming assistants: Successes and challenges," in *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering (ICSE)*, ACM, 2024.

[12] H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, and R. Karri, "Asleep at the keyboard? Assessing the security of GitHub Copilot's code contributions," in *2022 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2022, pp. 754–768.

[13] C. R. Prause, "Reputation-based self-management of software process artifact quality in consortium research projects," in *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering (ESEC/FSE)*, ACM, 2011, pp. 380–383.

[14] N. Raman, M. Cao, Y. Tsvetkov, C. Kästner, and B. Vasilescu, "Stress and burnout in open source: Toward finding, understanding, and mitigating unhealthy interactions," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, ACM, 2020, pp. 57–60.

[15] I. Shumailov, Z. Shumaylov, Y. Zhao, N. Papernot, R. Anderson, and Y. Gal, "AI models collapse when trained on recursively generated data," *Nature*, vol. 631, no. 8022, pp. 755–759, 2024.