An Empirical Study on Developers' Shared Conversations with ChatGPT in GitHub Pull Requests and Issues

Tongwei Zhang
t98zhang@uwaterloo.ca
University of Waterloo
Waterloo, Canada

1 Problem to be Solved:

This paper investigates how software developers actually use and share ChatGPT conversations in collaborative development workflows, particularly in GitHub Pull Requests (PRs) and Issues. The motivation is that while foundation models like ChatGPT are increasingly used for programming tasks, there is limited understanding of how these interactions shape collaboration, documentation, and communication among developers. The authors aim to fill this gap by systematically analyzing developers' shared ChatGPT conversations to understand what kinds of tasks are asked, how multi-turn dialogues evolve, and why and where developers share such links.

2 What is the New Idea

The authors create a dataset called **DevGPT**, which includes 580 initial prompts and 189 multi-turn conversations shared in PRs and Issues. They conduct qualitative coding to answer three research questions:

- RQ1: What types of inquiries developers present to ChatGPT.
- RQ2: How multi-turn conversations evolve and what roles different prompts play.
- RQ3: Why and where developers share ChatGPT links and who posts them.

They identify 16 task types (such as code generation, debugging, conceptual understanding, and documentation), 7 prompt roles (initial task, refinement, follow-up, confirmation, etc.), and 6 conversation flow patterns. For RQ3, they find that developers mainly share ChatGPT links to present potential solutions, document implemented solutions, or justify decisions. They also analyze where these links appear (e.g., issue descriptions, PR comments) and whether they are posted by authors or reviewers.

3 Positives:

- **Novel dataset and topic.** This is the first study focusing on how developers share ChatGPT outputs in real collaborative settings, which fills an important research gap.
- **Clear taxonomy and systematic coding.** The 16 task types and 7 prompt roles are well defined and supported with examples, giving a solid framework for future work.
- **Methodological rigor.** The authors ensure coding reliability with inter-rater agreement and provide transparent data collection and annotation procedures.
- **Practical implications.** The results can guide better design of AI-assisted developer tools and improve our understanding of how AI fits into real-world workflows.

4 Negatives:

- **Limited data scope.** The dataset covers only GitHub PRs and Issues up to late 2023, which may not represent all collaboration contexts or private repositories.
- **Mainly descriptive.** The study does not measure impact on productivity or code quality, focusing instead on classification and patterns.
- **Potential annotation bias.** Even though agreement is reported, qualitative coding may still contain subjective interpretations.
- **Lack of diversity in data sources.** Most conversations come from English repositories and may miss cultural or linguistic differences.

5 Future Work:

- **Build benchmarks from real conversations.** Construct realistic test sets that include multi-turn, ambiguous, or incomplete prompts inspired by observed patterns.
- **Develop role-aware collaboration tools.** Integrate ChatGPT provenance into developer platforms so reviewers and authors can trace how AI-generated suggestions evolved.
- **Quantitative outcome studies.** Analyze whether shared ChatGPT conversations correlate with better code quality or shorter review time.
- **Extend to industrial datasets.** Compare open-source and enterprise uses of ChatGPT in collaborative workflows.

6 Rating:

Rating: 4/5. This paper provides strong descriptive insights and solid empirical evidence on a new and important phenomenon—the social use of ChatGPT in open-source collaboration. The study is well structured and clearly reported, with transparent coding processes and meaningful visualizations. While the work could be strengthened with longitudinal data or outcome-based validation, it already contributes valuable foundational understanding to researchers and tool designers interested in AI-assisted software development.

7 Discussion Summary

7.1 Should PR templates include a "ChatGPT provenance" field?

7.1.1 *Benefits of including chat links.*

- Traceability helps reviewers understand decision-making processes and evaluate whether developers performed adequate due diligence or simply accepted the first AI suggestion

- Provides missing context that helps reviewers understand seemingly questionable implementation choices

7.1.2 *Concerns about this approach.*

- Linking ChatGPT conversations may implicitly suggest the AI output is correct, creating an inappropriate appeal to authority
- A better alternative would be for AI tools to provide references to human-written code sources (like Stack Overflow), which could be cited directly instead
- Developers are unlikely to genuinely use ChatGPT as an authoritative source, and claiming "ChatGPT told me to" could reflect poorly on them
- Experienced developers view ChatGPT as a collaborative tool rather than an authority
- LLMs are typically used in two scenarios: when correctness validation isn't required, or when alternative validation methods are available
- Developer responsibility for their code remains unchanged regardless of AI involvement—the final code is always the developer's responsibility

7.2 Ethics and Intellectual Property Risks

7.2.1 *Privacy and data exposure.*

- Information shared with ChatGPT should not be considered highly private, as the platform may use it for training purposes
- However, non-public repository content could still present confidentiality issues
- Anonymization tools exist to protect sensitive information before sending inputs to LLMs

7.2.2 *Legal and liability concerns.*

- ChatGPT conversations could become problematic evidence in lawsuits, potentially revealing inadequate risk analysis (similar to incriminating emails)
- Companies may be unaware of developers using ChatGPT, creating compliance gaps
- Real case: HR intervention occurred when developers used ChatGPT to discuss potentially unpublished content, raising concerns about training data usage and legal exposure

7.2.3 *Safety-critical application boundaries.*

- Critical applications still require traditional rigorous approaches
- Determining where to draw the line for extra caution is challenging and historically defined only after failures occur
- Historical examples demonstrate catastrophic consequences of insufficient caution: a radiation therapy device software change removed essential hardware safeguards, resulting in patient deaths; a rocket exploded due to 16-bit overflow when faster engines were installed without corresponding software updates

8 Other Comments

I think this is a very solid and interesting paper. The qualitative coding is careful, and the figures showing conversation flows are informative. It helps me understand that developers often use ChatGPT as a collaborative knowledge source rather than just a personal assistant. This study will be a useful reference for future research about integrating LLMs into real development workflows.