Open-Prompt: Towards Reproducible Agent-Driven Software Development

Anonymous Author(s)

Abstract

ACM Reference Format:
Anonymous Author(s). 2025. Open-Prompt: Towards Reproducible Agent-Driven Software Development. In *Proceedings of Workshop on Hot Topics in Operating Systems (HotOS '25).* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

As model capabilities continue to improve and programmer acceptance steadily grows, coding agents have advanced rapidly. Many software projects are already developed and maintained with coding agents, sometimes even without a human reviewing the codebase. The development workflow around coding agents is evolving quickly, and vibe coding is being adopted ever more widely.

However, because everyone is still in an exploratory phase, the ways people use coding agents vary widely [1], and recommended best practices for how to use coding agents most effectively are still being worked out. For instance, the prompt—a crucial resource in the development workflow—has no settled convention for how it should be managed.

Just as in traditional development most of a programmer's effort is translated directly into source code, in vibe coding the developer's intent and energy are translated directly into prompts. Indeed, compared with the large volume of generated code that has not yet been reviewed and confirmed, the prompt is a more distilled and more original description of the developer's intent, and carries high value. However, these prompts are rarely published or shared—even in open-source projects. More unfortunately, project developers themselves often do not preserve these high-value prompts; the agent sessions and histories used during development are frequently lost, creating additional difficulty for longer-term refactoring and for collaboration based on prompt sharing.

This paper advocates the importance of open-prompt, which is the practice of maintaining and sharing the prompts used during agent-driven software development. Specifically, our contribution is three-fold:

(1) We argue that open-prompt, as shown in Figure 1, is a natural extension of open-source into the agent era.
(2) We discuss why open-prompt significantly improves reproducibility, benefits the long-term development and maintenance of software, and encourages community collaboration and ecosystem building.
(3) Finally, we identify some of the challenges that open-prompt may face, and give some early explorations and tool design to make open-prompt more accessible.

Figure 1: TODO: From open-source to open-prompt.

2 From Open-Source to Open-Prompt

In the early days, programs existed directly as instructions in a binary; after compilers were developed, people began writing programs in high-level languages and relying on compilers to translate those high-level languages into the binary for the target platform. Open-source asks developers to release not only the binary but also the source code that precedes compilation, in order to encourage reuse and community collaboration.

Recently, after the rise of coding agents, people have begun describing development requirements in natural language; the coding agent translates that natural language into a high-level language, which is then translated by the compiler into the binary for the target platform. The development workflow has gained one extra step.

In the narrow sense, if open-source refers specifically to the act of releasing source code, then the practice of releasing the prompt that generated the source code calls for a new term. In this paper, we define open-prompt as the act of releasing the prompts used during a coding-agent-driven development workflow.

In the broad sense, if open-source points more toward the intent of providing the original traces of development and

HotOS '25, Banff, AB, Canada

ACM ISBN 978-1-4503-XXXX-X/2025/05
https://doi.org/XXXXXXX.XXXXXXX

preserving the developer's most original intent and effort, then open-prompt can be regarded as a special case of open-source in the era of coding agents.

When comparing open-source and open-prompt, a natural question arises: when the code can already fully reflect the development logic, do we still need open-prompt? Since a coding agent can read an open-source project directly, does that mean using code as the source of truth is already good enough?

While a coding agent can indeed continue development without the previous prompts—as shown by SWE-bench, an agent can submit a PR—whether this is optimal is not clear. There are three general concerns:

(1) *Performance degrades across iterations.* There is evidence [2] that iterative development can sometimes lead to a decline in code quality. Compared with repeatedly modifying the requirements, describing the complete system in one pass sometimes yields more concise code. An open-prompt project makes it simpler to refactor the project directly at the prompt level, rather than having to recover the original development intent or to build on top of it.
(2) *Lack of readability.* In coding-agent-driven development, an instruction of one or two sentences can often result in thousands of lines of code being added or removed. Compared with reading the code changes, reading the developer's prompt directly is more efficient for both humans and agents.
(3) *Loss of development context.* During the original development, the coding agent may have been instructed to avoid a particular design or dependency, and such a preference is not necessarily recorded by the agent. Even when a readme file summarizes the project, not all context is necessarily captured. In follow-on development, ignoring this context often means the developer has to repeat the same instructions.

Open-prompt proposes a strict but beneficial requirement for making the development workflow public. The various advantages of open-source also take on new interpretations in the open-prompt context, for example:

(1) As mentioned earlier, open-prompt can improve a project's readability, helping users and agents understand the developer's original intent and preserving context.
(2) Just as open-source gives external developers the right to review and refactor the code, open-prompt allows external developers to customize and regenerate the project at the semantic level, improving transparency and trustworthiness.
(3) Open prompts also improve the reusability of modules and designs, which can make community building and ecosystem construction easier. For example, prompts can be shared across projects with common design through a license, or interface-consistent software can be developed through a shared protocol document.

Given that sharing prompts has so many benefits, why is this practice currently relatively limited? In the next section, we concretize the practice of open-prompt, and while presenting its various challenges we discuss potential solutions.

3 Practicing Open-Prompt

In the previous section, we described open-prompt simply as the act of releasing prompts, but many details are worth discussing: which prompt content should be shared? What format should be adopted? How can reproducibility be improved? In this section, we first revisit a typical coding-agent-driven development workflow. Building on this, we discuss the various challenges that arise in implementing open-prompt, and give some simple potential designs as solutions.

First, we distinguish among different kinds of AI-assisted software development, ordered roughly from least to most AI usage into three categories:

(1) The case with the least AI usage is using only completion. The overall structure is still written by a human.
(2) Next is using AI interactively to generate large blocks of code or complete files, while the human still directly steps in to debug and fix errors.
(3) Driving a coding agent entirely through natural language.

References

[1] 2025. How People Use Coding Agents. arXiv preprint arXiv:2509.17548. https://arxiv.org/pdf/2509.17548. TODO: verify author/title/year..
[2] 2026. Code Quality Degradation under Iterative Agent Development. arXiv preprint arXiv:2603.24755. https://arxiv.org/pdf/2603.24755. TODO: verify author/title/year..