Kourai Khryseai: Transparent Human-on-the-Loop Multi-Agent Software Development

Arnaldo Barea
Rochester Institute of Technology, Rochester, NY, USA
ajb6289@rit.edu

Abstract

Multi-agent coding systems are increasingly capable, but many still hide coordination decisions from users. We present **Kourai Khryseai** (**Kourai**), an interactive software development environment that treats multi-agent work as a supervised collaboration problem. An orchestrator routes requests to specialist agents for planning, coding, testing, style review, and commit synthesis, while agents stream intermediate status and request clarification when requirements are ambiguous. The backend combines independent A2A-connected services, MCP-based tool use, shared local SQLite state, and end-to-end tracing through OpenTelemetry, Jaeger, and Prometheus. The same orchestration layer powers a CLI, desktop GUI, and visual-novel-style interface, enabling studies of interface and embodiment without changing backend logic. As a systems artifact, Kourai includes tests for pipeline ordering, context accumulation, clarification loops, bounded repair behavior, and graceful degradation under specialist failure.

Keywords: AI agents, multi-agent systems, software engineering, human-on-the-loop, observability

1. INTRODUCTION AND RELATED WORK

Recent work on agentic software engineering includes issue-resolution benchmarks and developer-facing platforms such as SWE-bench and OpenHands [3, 5]. Multi-agent orchestration is studied in systems such as Magentic-One [2], while human-in-the-loop software agents are treated directly in HULA [4] and agent observability in AgentTrace [1]. Kourai sits at this intersection: a software engineering system with explicit orchestration, streamed intermediate state, and interactive supervision. We use *human-on-the-loop* deliberately: the user supervises and redirects specialist agents rather than authoring each step, distinguishing Kourai from human-in-the-loop agents such as HULA.

2. SYSTEM DESIGN

Kourai is organized as an orchestrated specialist pipeline. Hephaestus routes tasks to Metis for planning, Techne for coding, Dokimasia for testing, Kallos for style review, and Mneme for change summarization. Agents are independent HTTP services communicating through A2A, while filesystem, shell, git, and documentation access are exposed through MCP servers. Conversation history and agent state are stored locally in SQLite. When a decision is ambiguous, the pipeline pauses for user input; when review finds issues, the system runs a bounded repair loop instead of silently retrying indefinitely. Our emphasis is not autonomous benchmark maximization, but making coordination legible through streamed status and end-to-end tracing [1] during real development work.

3. ARTIFACT EVALUATION

We evaluate the prototype as a systems artifact rather than a benchmark submission. The repository includes unit and integration tests covering end-to-end pipeline ordering, accumulation of intermediate context across specialists, clarification loops, iterative Techne–Kallos repair, and graceful degradation when a specialist is unreachable. The same backend supports three hosts: a terminal client, a desktop GUI, and a Ren'Py visual-novel interface. This lets interface and embodiment vary while the orchestration logic remains fixed.

4. CONCLUSION

Kourai suggests that transparency is not only a user-experience choice but also a systems property. Explicit specialization improves modularity and debuggability, while streamed coordination and human-on-the-loop checkpoints make agent behavior easier to inspect and redirect. We see the artifact as a useful platform for future studies of agent memory, orchestration, debugging, and user trust in software engineering agents. Project documentation is available at https://ajbarea.github.io/kourai-khryseai/, with source code at https://github.com/ajbarea/kourai-khryseai. The author thanks Dr. Leon Reznik for advising this work.

References

[1] A. AlSayyad, K. Y. Huang, and R. Pal. Agenttrace: A structured logging framework for agent system observability. *arXiv preprint arXiv:2602.10133*, 2026.

[2] A. Fourney et al. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*, 2024.

[3] C. E. Jimenez et al. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024.

[4] W. Takerngsaksiri et al. Human-in-the-loop software development agents. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2025.

[5] X. Wang, B. Li, Y. Song, et al. OpenHands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2025.