CS848 Week 3 Review

Zhiheng Lyu
z63lyu@uwaterloo.ca
University of Waterloo
Waterloo, ON, CA

ACM Reference Format:
Zhiheng Lyu. 2025. CS848 Week 3 Review. In . ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Paper 1: Human-in-the-Loop LLM Agents for Software Development

- **Problem Being Solved.** Large language model (LLM) agents have shown strong performance on structured benchmarks such as SWE-bench, yet their utility in real-world software engineering remains uncertain. Benchmarks tend to present problems in a sequential, waterfall-like form: a clear issue description, a well-defined repository, and a fixed test harness. In contrast, enterprise environments like Atlassian JIRA are messy. Issues are short, ambiguous, and distributed across heterogeneous repositories and languages. Moreover, both clients and programmers often do not know what they want until they see code in action. This iterative discovery process is central to software development, raising the question: can LLM-based agents actually support this human feedback loop at scale, or do they falter when faced with noisy, evolving requirements?
- **New Idea.** The paper introduces HULA (Human-in-the-Loop LLM-based Agents), a framework that embeds AI agents directly within JIRA workflows. HULA operates through three roles:
  (1) An *AI Planner*, which identifies relevant files and drafts a plan of action.
  (2) An *AI Coding agent*, which generates proposed code modifications.
  (3) A *Human Engineer*, who reviews, edits, and approves at each step.

  This tripartite structure reflects an explicit rejection of full autonomy. Instead, HULA positions the AI as a junior collaborator, capable of drafting but dependent on human oversight. The evaluation was multi-stage: offline benchmarking (SWE-bench and 369 JIRA issues), an online deployment over 663 live issues, and a survey of 109 Atlassian engineers.
- **Positive Points.**
  - *Real-world deployment.* By moving beyond controlled benchmarks and integrating into live JIRA workflows, the study demonstrates adoption at scale. This is a rare and valuable contribution in a field dominated by synthetic datasets.
  - *Multi-modal evaluation.* The triangulation of offline tasks, online deployment, and user surveys strengthens confidence in the findings. It avoids relying solely on pass@k metrics or simulation-based assessments.
  - *Human-centered workflow.* The design explicitly incorporates human review at every stage, acknowledging that developers must validate, adapt, and refine code. Many survey participants reported faster onboarding and acceleration of routine tasks.
  - *Evidence of practical utility.* Plan approval rates exceeded 80%, and over 50 pull requests were merged. These numbers indicate that AI can provide incremental value, even if the code is not fully correct.
- **Negative Points.**
  - *Shallow problem framing.* While the paper acknowledges the gap between benchmarks and enterprise complexity, the analysis remains surface-level. For instance, recall for file localization drops from 86% on SWE-bench to 30% in JIRA, yet the authors stop short of dissecting the root causes (e.g., short issue length, missing context, multi-repo structures). Without such depth, the work risks being classified as "low-hanging fruit": an early exploration that demonstrates feasibility but leaves most challenges unresolved.
  - *Economic value underexplored.* The paper emphasizes drop-off percentages and merge success, but ignores the cost of human review. Developers must spend significant cognitive energy editing, correcting, or discarding AI-generated code. Without factoring in this human effort, efficiency claims are incomplete. From a productivity perspective, the true metric should be net savings in developer time, not raw AI contribution.
  - *Metrics misaligned with practice.* The discussion highlighted that drop-off percentages (shown in graphs) may not matter as much as the authors assume. In practice, a higher number of merge requests (MRs) can be a healthy signal, indicating many opportunities for incremental improvement. What matters more is how developer energy is distributed across stages: are they spending time on trivial edits, or are they freed to focus on higher-value challenges?
  - *Moderate technical novelty.* The architecture largely orchestrates GPT-4 into a multi-stage pipeline. While valuable as deployment evidence, it lacks algorithmic or methodological innovation (e.g., adaptive retrieval, feedback-aware training).
- **Future Work.**
  - *Integrate richer context.* Go beyond short JIRA issue text by linking documentation, historical tickets, commit histories, and repository embeddings. This could narrow the gap between enterprise tasks and benchmark-style clarity.
  - *Redefine evaluation metrics.* Replace or augment drop-off percentages with measures that reflect real value: maintainability, readability, defect density, and, crucially, net developer time saved.

Conference'17, Washington, DC, USA
2025. ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/10.1145/nnnnnnn.nnnnnnn

  - *Model economic trade-offs.* Explicitly quantify the cost of human review at each pipeline stage. Only by comparing AI contribution against human editing burden can we evaluate whether the workflow is cost-effective.
  - *Continuous feedback integration.* Move beyond simple approval/rejection to structured annotations and adaptive retraining. This could allow systems like HULA to learn iteratively, reducing human review effort over time.
  - *Compare paradigms.* Systematically evaluate HULA against AI-native IDEs (e.g., Cursor, Copilot) to see which human-AI interfaces developers prefer and why.
- **Rating.** I would rate the paper 3.5/5. It represents an important step toward bridging benchmarks and enterprise practice, but its contributions lie more in deployment logistics than in deep technical innovation. The lack of economic modeling and the superficial treatment of performance gaps mean that its conclusions could shift substantially with further optimization. The study is best seen as a first demonstration of feasibility, not as definitive evidence of sustainable productivity gains.
- **Discussion Points.**
  - Should AI-assisted development aim to automate more aggressively, or double down on iterative human-AI co-design that embraces feedback as essential?
  - How do we evaluate economic value realistically? What metrics should account for human review effort, cognitive load, and opportunity cost?
  - Are "low-hanging fruit" tasks sufficient to justify enterprise integration, or do they risk creating the illusion of efficiency while masking deeper bottlenecks?
  - Could a surge in merge requests (MRs) actually be positive, signaling more iteration and refinement, rather than inefficiency?