

AI-POWERED AUTONOMOUS WORKFORCE ASSISTANTS

Presented by:

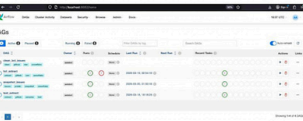
Rutuja Kadam
Shatayu Thakur
Shreyas Mohite
Shubham Naik

Project Advisors:

Dr. Simon Shim
Dr. Lee C. Chang

Introduction

Every sprint, engineering teams that manage sizable open-source repositories lose hours due to the same unseen issue: problems accumulate, PRs stop, and by the time a blocker is discovered, a deadline has already passed. Every sprint, the manual triage procedure takes hours and continues to overlook items. The slowest 10% of issues on Apache Airflow alone take more than 22 days to resolve, yet the median closes in a single day. This skew is so severe that traditional alerting criteria completely fail. A Slack-based bottleneck detector and a VSCode autonomous patch planner five refined models with human validation at every stage are two closely integrated AI features developed by AutoBot based on 12,719 actual GitHub issues and 43,731 pull requests.



Methodology

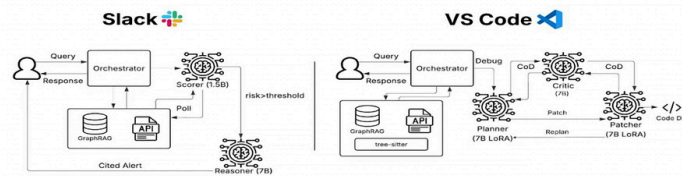
About 56,000 problems and PRs from apache/airflow were used to extract 14 GitHub API signal types using async ETL. Temporally-honest training snapshots were created at project-relative percentiles (P50=1d, P75=5d, P90=22d, and P95=44d), and all closure signals were removed before to training.

Every 30 minutes, the Slack Bot orchestrator uses a Scorer to Reasoner pipeline to identify bottlenecks. It then routes high-risk issues as plain-language alerts straight to the team's Slack channel.

Meanwhile, the VSCode Plugin drives a Planner - Patcher - Critic loop with human approval gates at every stage, ensuring that nothing enters the main branch without developer approval.

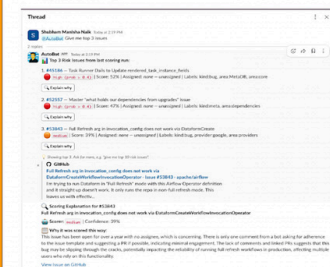
This dual-model architecture ensures that critical bottlenecks are addressed with machine precision while maintaining a compact, cost-effective inference footprint. Ultimately, this integration accelerates engineering throughput by reducing the manual overhead of triage for complex, large-scale repositories.

Analysis and Results



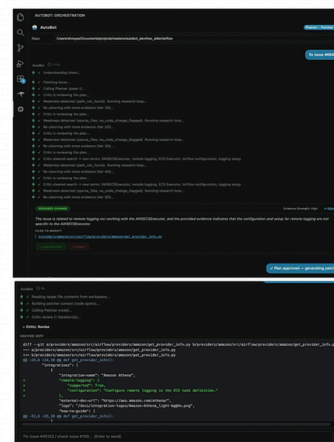
Bottleneck Detector:

Every 30 minutes, the Scorer sweeps across all open problems, identifying behavioral cues that humans overlook at scale, such as blocked PRs, assignee changes, comment gaps, and CI failures. It then gives each issue a calibrated risk float that indicates if it requires immediate attention. Every issue that the Scorer flags as high-risk is translated by the Reasoner into a two to three sentence plain-language briefing that is sent straight to the team's Slack channel. This briefing tells the scrum master exactly what is stalling, why it matters, and how urgent it is without requiring them to open a single GitHub tab.



Autonomous Patch Planner:

To determine precisely which files and functions need to be changed, the Planner examines both the problem and the actual codebase structure. This ensures that no paths are misdirected or hallucinated by basing each recommendation on the actual repository. Using the Planner's results, the Patcher creates a real unified diff that targets the precise function spans found. This is a ready-to-apply code modification rather than a recommendation or explanation of what has to be done. The developer receives a patch that has already undergone self-review before they see it because the Critic evaluates the generated patch and provides an ACCEPT, REVISE, or REJECT decision with specific line-level reasoning. This process drives an iterative retry loop that detects bugs and architectural violations before CI even runs.



Summary/Conclusions

Summary

AutoBot is a multi-model agentic system that automates software delivery on apache/airflow through two pipelines: a Slack bot that scores open issues for delivery risk and pushes plain-language briefings to stakeholders, and a VS Code extension that autonomously plans, generates, and validates code patches with human-in-the-loop approval. Five LoRA-adapted models (Scorer, Reasoner, Planner, Patcher, Critic) are trained on 12,000+ issues and 41,000+ PRs, backed by hybrid retrieval combining GraphRAG co-resolution patterns with Tree-sitter structural indexing.

Conclusions

- Hybrid retrieval (GraphRAG + Tree-sitter) outperforms keyword heuristics, historical co-resolution patterns find relevant files while symbol-level spans keep context bounded.
- Deterministic orchestrator refinement beats unconditional multi-pass prompting: remaining Planner errors are recall misses best fixed by enriching evidence, not adding LLM calls.
- Live RLHF via Slack reactions provides a zero-effort annotation signal, enabling weekly DPO retraining without dedicated labeling infrastructure.

Key References

- [1] Jimenez et al. (2024). SWE-bench: Can language models resolve real-world GitHub issues? ICLR 2024.
- [2] Xia et al. (2024). Agentless: Demystifying LLM-based software engineering agents. arXiv:2407.01489.
- [3] Hu et al. (2021). LoRA: Low-rank adaptation of large language models. arXiv:2106.09685.
- [4] Sayagh et al. (2024). What makes a GitHub issue ready for Copilot? arXiv:2512.21426.
- [5] Yang et al. (2024). Qwen2 technical report. arXiv:2407.10671.

Acknowledgements

The authors thank Dr. Simon Shim and Dr. Lee C. Chang, Department of Applied Data Science, San Jose State University, for their guidance throughout the project lifecycle.

Interested in partnering with us on a capstone project or exploring other ways to collaborate with the College of IDS? Contact CIDS_CAP@sjsu.edu to discuss potential projects and partnership opportunities.