

WhoBrokeMyTest: An AI Agent for Test Failure Analysis

Cristina Gavrilă*
cristina.gavrila@siemens.com
Siemens Industry Software S.R.L.
Braşov, Romania

Ioana Lăzărescu
ioana.lazarescu@siemens.com
Siemens Industry Software S.R.L.
Braşov, Romania

Raluca Daicu
raluca.daicu@siemens.com
Siemens Industry Software S.R.L.
Braşov, Romania

Abstract

This paper presents WhoBrokeMyTest, an innovative AI agent developed to address a significant challenge in software development: the time-consuming process of investigating automated test failures. Identifying the specific code changes responsible for test failures consumes substantial developer resources. Our solution leverages the agent approach to Large Language Model (LLM) application development, combining reasoning capabilities with specialized tools that access code repositories, test results, and version control metadata. Initial implementation demonstrates that WhoBrokeMyTest can reduce investigation time from hours to minutes, with projected savings of approximately 10 developer days per release cycle per team. This significant improvement in productivity allows development teams to reallocate resources from maintenance tasks to feature development. Our work contributes to the growing field of LLM-powered software development tools aimed at optimizing the software testing lifecycle through intelligent agent frameworks.

CCS Concepts

• **Software and its engineering** → **Software testing and debugging**; • **Computing methodologies** → **Artificial intelligence**.

Keywords

AI agents, software testing, continuous integration, automated test failures, LLM applications, developer productivity

ACM Reference Format:

Cristina Gavrilă, Ioana Lăzărescu, and Raluca Daicu. 2025. WhoBrokeMyTest: An AI Agent for Test Failure Analysis. In *Proceedings of womENCourage 2025 Computer Science: ACM Celebration of Women in Computing (womENCourage '25)*. ACM, New York, NY, USA, 2 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 Introduction and Problem Statement

Modern software development relies on continuous integration and automated testing, but as codebases grow, the volume of daily tests creates challenges when failures occur. In large-scale development environments, there are thousands of automated tests, which

*All authors contributed equally to this research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

womENCourage '25, Brasov, RO

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/2018/06
<https://doi.org/XXXXXXX.XXXXXXX>

cannot be run after each code change, so they run once per day. Developers often spend hours investigating root causes through logs and code changes across multiple branches—a time-consuming process that diverts resources from feature development. These manual inspection approaches have become increasingly inadequate for today's development velocities.

This paper introduces WhoBrokeMyTest, an innovative AI agent specifically designed to address the test failure investigation challenge. Unlike conventional approaches that provide static analysis or simple automation, WhoBrokeMyTest employs an agent architecture that combines reasoning capabilities with specialized tools by leveraging recent advancements in Large Language Models (LLMs) and agent-based architecture patterns [1].

This agent-based architecture enables WhoBrokeMyTest to perform complex analyses by orchestrating multiple operations: retrieving branch histories, examining modified files, analyzing error patterns, and correlating changes with test failures. The agent can pinpoint the exact branch and modified files that triggered test failures and proactively recommend relevant tests to run based on uncommitted file modifications, thereby preventing potential regressions.

The name WhoBrokeMyTest was deliberately chosen to reflect the tool's end-to-end accountability tracking in the software development lifecycle. While the tool does identify code changes responsible for test failures, the 'Who' aspect aligns with the principles of ownership and responsibility. When a test fails, it is not just about identifying the technical cause but also about establishing clear ownership of the fix. Once the author is identified, they are contacted to provide a solution.

2 Implementation

WhoBrokeMyTest has been implemented as a modular system designed to interact with modern development and testing environments. It operates exclusively within Siemens' internal network infrastructure, where all data processing occurs on internal servers without any external API calls.

The system has been specifically designed and tested for use with Siemens' proprietary source code, ensuring that sensitive information remains within the corporate environment at all times. While the underlying concept could be adapted for open-source projects, the current implementation is exclusively used within Siemens' development workflows, demonstrating its effectiveness in handling proprietary software development needs.

This implementation demonstrates the broader potential of LLM-based systems in automating aspects of the software testing lifecycle, from understanding test failures to supporting results interpretation and analysis [5]. The agent interface allows developers to query the system about test failures using conversational prompts.

The implemented system leverages Azure OpenAI's GPT-4o for natural language understanding, integrates LlamaIndex [4] as the agent orchestration framework with semantic search capabilities, and employs Streamlit (<https://streamlit.io>) for the web-based chat interface. On top of this foundation, the agent incorporates the following specialized tools, enabling users to query information about test failures and associated branches:

- Tool for retrieving current date
- Tool for retrieving information about pushed branches in a certain time interval or a given day
- Tool to retrieve information about automated tests
- Tool to retrieve information about modified files in the repository
- Tools for asking the user about specific information if not given, like the time interval or error message: purpose is to instruct the agent to ask for very specific information in order to give the correct answer instead of guessing

3 Capabilities and Performance

This architecture enables WhoBrokeMyTest to perform time consuming operations including:

- Retrieving and analyzing complete branch histories to identify the potential test failure culprits
- Examining modified files across multiple commits and branches
- Correlating specific code changes with test failures through semantic analysis
- Pinpointing the branch and modified files responsible for test failures
- Proactively recommending relevant tests to run based on uncommitted file modifications

Based on time logging data collected over 8 months from a team of 6 developers, we measured that investigating test failures originating from other teams consumed approximately 8 working days. These measurements specifically tracked time spent on cross-team test failure investigations, as within-team failures are typically identified and resolved more quickly through direct communication. Extrapolating this data to our standard 12-month release cycle and accounting for vacation periods, we estimate approximately 10 developer days are spent per release cycle on such investigations. WhoBrokeMyTest can reduce this investigation time by approximately 95%, bringing the typical investigation time from 2-3 hours down to 5-10 minutes per incident, which translates to reducing the 10-day overhead to approximately 0.5 days per release cycle.

4 Current limitations

Despite its promising results, WhoBrokeMyTest currently has several limitations:

- Occasional hallucinations, leading to providing test names that do not exist
- Inaccurate branch identification in some test failure scenarios
- Incomplete retrieval of relevant tests
- Limited effectiveness unless specific keywords are present in the error message

- Reliance on textual similarities between the error message and repository elements (commit messages, branch names, or modified filenames)

5 Future Improvements

To address these limitations and enhance capabilities, the plan is to:

- Create scheduled tasks that update the repository history index daily
- Refine retrieval prompts to optimize repository history search efficiency
- Enhance branch analysis with additional metadata filtering capabilities
- Develop automatic test log retrieval for failed test cases
- Expand detection capabilities for tests failing due to numerical differences
- Implement direct tool response mechanisms to reduce hallucinations and generative inaccuracies
- Optimize local file modification detection algorithms
- Design scalable architecture for production environments supporting multi-user access and usage analytics
- Extend capabilities to include automated fix suggestions for common test failure patterns, aligning with emerging autonomous AI agent frameworks [2]

6 Conclusion

WhoBrokeMyTest demonstrates a promising application of AI agent technology to concrete software development challenges. By combining the reasoning capabilities of LLMs with specialized tools and domain-specific knowledge, the agent dramatically reduces the time and effort required for test failure investigation. This approach not only improves immediate developer productivity but also enables strategic reallocation of resources from maintenance tasks to innovation and feature development.

As development environments grow increasingly complex, agent-based solutions like WhoBrokeMyTest demonstrate how AI can be effectively applied to tackle specific, high-value challenges in the software development lifecycle. [1, 3].

References

- [1] Robert Feldt, Sungmin Kang, Juyeon Yoon, and Shin Yoo. 2023. Towards Autonomous Testing Agents via Conversational Large Language Models. arXiv:2306.05152 [cs.SE] <https://arxiv.org/abs/2306.05152>
- [2] Mohamed Amine Ferrag, Norbert Tihanyi, and Merouane Debbah. 2025. From LLM Reasoning to Autonomous AI Agents: A Comprehensive Review. arXiv:2504.19678 [cs.AI] <https://arxiv.org/abs/2504.19678>
- [3] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2024. Large Language Models for Software Engineering: A Systematic Literature Review. arXiv:2308.10620 [cs.SE] <https://arxiv.org/abs/2308.10620>
- [4] Jerry Liu. 2022. *LlamaIndex*. doi:10.5281/zenodo.1234
- [5] Betim Sherifi, Khaled Slhoub, and Fitzroy Nembhard. 2024. The Potential of LLMs in Automating Software Testing: From Generation to Reporting. arXiv:2501.00217 [cs.SE] <https://arxiv.org/abs/2501.00217>

Received May 2025; revised 2025; accepted 2025