Graph-Based LLM Prompting for Scalable Microservice API Testing

Md Arfan Uddin
Systems and Industrial Engineering
University of Arizona
Tucson, Arizona, USA
arfan@arizona.edu

2025 IEEE International Conference on Service-Oriented System Engineering (SOSE) | 979-8-3315-8911-0/25/$31.00 ©2025 IEEE | DOI: 10.1109/SOSE67019.2025.00034

***Abstract*—Microservices offer flexibility and scalability, but their decentralized and fast-changing nature makes it difficult to maintain consistent and meaningful test coverage, particularly at the level of service endpoints. Services are developed independently, logic is spread across multiple layers, and execution paths vary widely based on input and control flow. As a result, automated testing is hard to scale, and manual testing is time-consuming and error-prone. Recent advances in large language models present a promising opportunity for generating tests automatically. However, existing approaches often rely on providing the entire source code as input, which can exceed model limitations and include unrelated logic that reduces test quality. This paper proposes a structured approach that uses interprocedural control flow graphs to guide language models in generating accurate, maintainable endpoint tests that better reflect the complexity of modern microservice systems.**

***Index Terms*—Microservice, API Testing, Test Automation, Large Language Model (LLM), Control Flow Graph (CFG), Static Analysis**

I. Introduction

Microservice architectures support agility and scalability by allowing independent development and deployment of modular services. Yet this independence introduces substantial testing challenges. Ensuring robust endpoint-level test coverage is especially difficult, as logic often spans multiple components and includes nested conditions or internal service calls.

Conventional test frameworks struggle to capture these complex behaviors, particularly in distributed environments. Recently, Large Language Models (LLMs) have shown promise for automating test generation from source code and descriptions [1]. However, full-source LLM prompts often exceed context limits and introduce noise, which reduces test quality and scalability [2]. This paper explores a more targeted alternative: using control-flow path representations to help LLMs generate precise, maintainable endpoint tests tailored to the microservice context.

II. Motivation and Problem Statement

Microservices evolve rapidly under decentralized development, making it difficult to achieve and maintain comprehensive API test coverage. Endpoint behavior is often defined by dynamic logic involving nested conditionals, method chaining, and cross-service flows—patterns that are hard to trace and test manually.

Recent work has demonstrated that intermediate representations (IRs) are useful in statically analyzing service logic, even without complete code access [3]. Ehsan et al. [4] identify key limitations in conventional endpoint testing, such as authentication barriers, hidden logic, and fragmented visibility. These gaps make scalable test automation especially difficult.

To overcome these issues, we propose using Interprocedural Control Flow Graphs (ICFGs) [5] to capture endpoint logic at the path level. ICFGs represent control flow through conditionals, loops, and nested calls in compact, semantically meaningful units. This leads to our central research question: **Can path-specific control flow graphs replace full-source input for scalable and effective endpoint test generation in microservice systems using LLMs?**

III. Vision

We envision a structured pipeline where interprocedural control flow graphs (ICFGs) serve as the bridge between static analysis and LLM-driven test generation for microservice API endpoints. Instead of feeding full source files—which are often large and cluttered with unrelated logic—our approach extracts individual control-flow paths that represent distinct execution behaviors of a given endpoint.

We use a tool like JavaParser to statically analyze endpoint implementations and generate ICFGs, where nodes represent methods, conditions, or loops, and edges capture call and branch relationships. This mirrors prior efforts in generating intermediate representations for service-level analysis [3]. Each graph path is serialized into a structured format—such as pseudocode or logical steps—for use in LLM prompts. A prototype implementation of this pipeline is currently under development and will be released as open source[2].

As illustrated in Figure 1, the process begins with a request (e.g., `/api/order/1`) to a microservice endpoint. The corresponding logic is statically analyzed and transformed into an ICFG that captures control flow across nested calls, conditionals, and loops. Each unique path—such as invalid input, resource not found, or successful response—is isolated and used to construct targeted LLM prompts via a prompt engineering layer.

[1]Uber 2016 Microservice Architecture [6].
[2]https://github.com/arfan-rfn/ms-testing-llm-icfg

2642-6587/25/$31.00 ©2025 IEEE
DOI 10.1109/SOSE67019.2025.00034

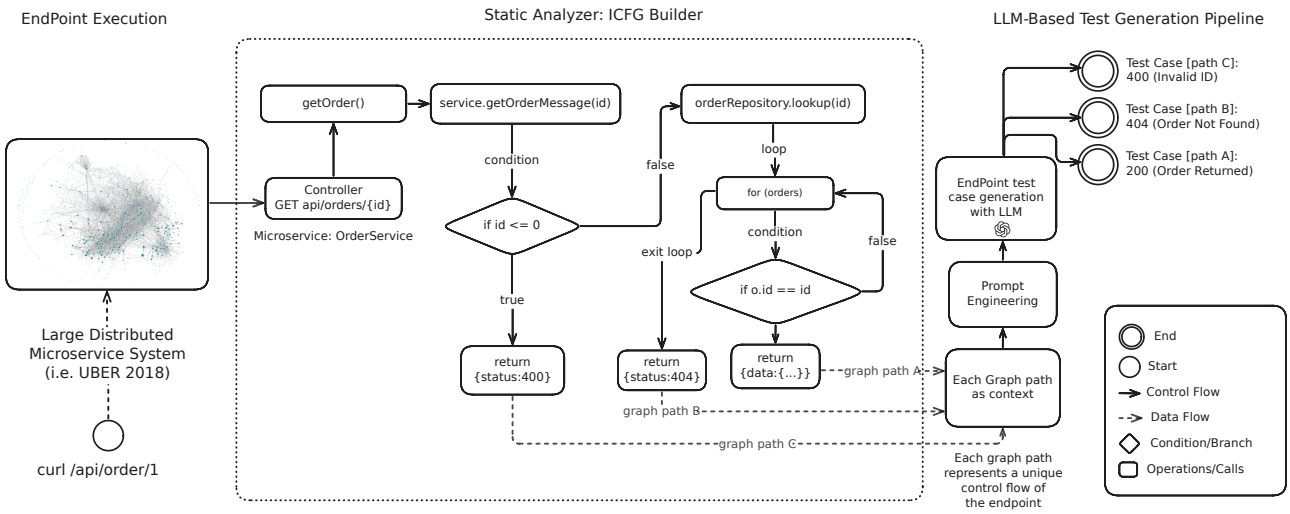

Fig. 1. LLM-Based API Test Generation Using ICFG Path-Specific Contexts.[1]

These prompts present the model with a bounded, semantically meaningful representation of one control path. The LLM then generates a corresponding test case (e.g., for HTTP status 400, 404, or 200), ensuring that distinct logical paths are explicitly covered. This modular approach improves scalability, maintains clarity, and enables test generation without revealing the full source code.

IV. Potential Benefits and Impact

The proposed ICFG-based test generation approach offers several advantages for microservice systems. Isolating execution paths enables more focused and relevant LLM inputs, improving the quality and accuracy of generated test cases. Unlike full-source prompts, this method scales effectively with large codebases, as each path is concise and self-contained. It also supports incremental test updates—only paths affected by changes need to be regenerated, improving maintainability over time. Because each control-flow path is explicitly modeled, the approach enables more thorough code coverage and helps identify untested or unreachable (dead) code. Additionally, abstracting endpoint behavior into ICFGs reduces the need to expose full source code, making the method suitable for secure or privacy-sensitive environments. Overall, it aligns well with the modular nature of microservices, offering a scalable and adaptable solution for automated endpoint testing.

V. Challenges and Future Directions

This paper presents a vision for scalable endpoint test generation using ICFG-guided LLM prompting. While promising, the approach introduces several challenges. Generating accurate ICFGs across diverse microservice implementations—especially those involving asynchronous behavior, dynamic dispatch, or third-party dependencies—requires mature static analysis tools. Representing these graphs in a form compatible with LLMs remains nontrivial, as models operate over sequential inputs rather than structured graphs. Evaluating test quality will also require benchmarks that capture coverage, correctness, and behavioral diversity. A prototype is currently under development, and future work will focus on validating its feasibility and effectiveness in real-world environments. Additional directions include enriching ICFGs with metadata such as API contracts and extending the pipeline to support cross-service integration testing.

References

[1] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, and Q. Wang, "Software testing with large language models: Survey, landscape, and vision," *IEEE Transactions on Software Engineering*, 2024.

[2] E. Quevedo, A. S. Abdelfattah, A. Rodriguez, J. Yero, and T. Cerny, "Evaluating chatgpt's proficiency in understanding and answering microservice architecture queries using source code insights," *SN Computer Science*, vol. 5, no. 4, p. 422, 2024.

[3] T. Cerny, G. Goulis, and A. S. Abdelfattah, "Towards change impact analysis in microservices-based system evolution," *arXiv preprint arXiv:2501.11778*, 2025.

[4] A. Ehsan, M. A. M. Abuhaliqa, C. Catal, and D. Mishra, "Restful api testing methodologies: Rationale, challenges, and solution directions," *Applied Sciences*, vol. 12, no. 9, p. 4369, 2022.

[5] A. Flexeder, B. Mihaila, M. Petter, and H. Seidl, "Interprocedural control flow reconstruction," in *Programming Languages and Systems: 8th Asian Symposium, APLAS 2010, Shanghai, China, November 28-December 1, 2010. Proceedings 8*, Springer, 2010, pp. 188–203.

[6] "Introducing domain-oriented microservice architecture." Accessed: 2025-05-30, Uber. (2020), [Online]. Available: https://www.uber.com/blog/microservice-architecture/.