University of São Paulo
Institute of Mathematics and Statistics Bachelor of Computer Science

Automated Documentation and Testing for Microservice Reusability

Agnaldo Nunes de Oliveira

MAC0499 — Capstone Project

Supervisor: Prof. Alfredo Goldman
Co-supervisor: MSc. João Francisco Lino Daniel

São Paulo
2025

Abstract

Microservice architectures (MSAs) offer numerous advantages in modern software development, including increased scalability and agility. However, maintaining reusability in MSA-based systems presents significant challenges, particularly in the areas of API documentation and integration testing. This project addresses these challenges by exploring the application of Artificial Intelligence (AI) techniques, specifically Large Language Models (LLMs), to automate the generation of API documentation and integration tests, thereby promoting service reusability and reducing manual effort.

This work is motivated by the observation that existing catalogs of microservice patterns and refactorings lack adequate tool support to facilitate their practical application. To bridge this gap, this project leverages AI for Software Engineering, focusing on the development and evaluation of an AI-powered system that automates the generation of comprehensive and accurate API documentation adhering to industry standards like OpenAPI. Furthermore, the system automates the generation of integration tests that effectively verify the reusability of services in different contexts, ensuring their reliability and compatibility.

The methodology involves a comprehensive literature review, system design and architecture, implementation and prototyping of a Minimum Viable Product (MVP), evaluation and refinement based on experimental results, and documentation and thesis writing. The system will be designed considering its integration as a black box solution or its linkage to an existing stack, and will explore the automation of code annotation for .yaml generation to facilitate documentation and integration. The implementation will leverage LLMs and Retrieval-Augmented Generation (RAG) techniques.

The expected outcome is a functional AI-powered system that demonstrates the feasibility of automated API documentation and integration test generation for promoting service reusability in MSA-based systems. The project will also contribute to a better understanding of the challenges and opportunities associated with applying AI to software engineering.

Table of Contents

Introduction

Microservices architecture (MSA) has become an increasingly popular approach for developing complex systems, offering advantages such as scalability, flexibility, and team independence (NEWMAN, 2015)[1]. However, adopting MSA brings significant challenges, especially regarding maintenance, documentation, and ensuring service quality.

One of the main bottlenecks in MSA-based systems is the need to keep API documentation up-to-date and accessible, as well as ensuring that integration tests reflect the actual behavior of services. This challenge is amplified in MSA due to the distributed nature of the architecture, where numerous independent services interact with each other through APIs. Lack of adequate documentation hinders service discovery and reuse, leading to duplicated efforts and increased system complexity. Similarly, the absence of automated and reuse-focused integration tests compromises system reliability and stability, making modifications riskier and more time-consuming

Despite the recognized benefits of MSA, the development and maintenance of these systems often face significant challenges due to the lack of adequate tooling. Manually creating and updating API documentation is a time-consuming and error-prone task, especially in large and complex systems with numerous microservices. This often leads to outdated or incomplete documentation, making it difficult for developers to understand and use the available services.

Similarly, the creation of integration tests that effectively cover the interactions between microservices is a complex and labor-intensive process, often resulting in insufficient test coverage and increased risk of integration issues. The absence of automated solutions for these tasks not only increases development costs but also hinders the agility and scalability of MSA-based systems.

In this context, service reuse emerges as a critical strategy for addressing the challenges of MSA. By promoting the reuse of existing services, organizations can reduce development costs, minimize redundancy, and ensure consistency across the system. However, effective service reuse requires not only well-documented and easily discoverable services but also robust integration tests that ensure the compatibility and reliability of reused components. Therefore, the automation of documentation and testing must be aligned with the goal of promoting and facilitating service reuse.

Faced with this scenario, this capstone project aims to support professionals in promoting and ensuring reusability during the development and maintenance of MSA-based systems. To achieve this goal, we will investigate and propose solutions

to automate the generation of API documentation and integration tests in microservices architectures, focusing on service reuse. We will explore the use of large language models (LLMs) and retrieval-augmented generation (RAG) techniques to create tools that facilitate the work of developers and ensure the quality and maintainability of systems.

We believe that automating documentation and testing, combined with the use of AI techniques, can significantly reduce the time and effort required to maintain a healthy and evolving microservices system, allowing teams to focus on adding value to the business.

Literature Review

Modifiability

Modifiability, as a key characteristic of robust software architectures, focuses on the system's capacity to absorb alterations with minimal impact. It goes beyond mere adaptability, encompassing the economic aspects of change - the resources, time, and potential disruptions incurred when modifications are necessary. A truly modifiable system anticipates change, embedding mechanisms that allow for seamless integration of new features, technologies, or adaptations to evolving requirements.

This attribute is not limited to code-level adjustments. It extends to the entire ecosystem of the software, including infrastructure, dependencies, and even the processes governing development and deployment. Changes can originate from various sources, ranging from developers implementing new functionalities to system administrators adjusting configurations for optimal performance.

Therefore, when considering modifiability, we must account for both the upfront investment in designing for change and the ongoing costs associated with implementing specific modifications. A well-designed system will minimize both, allowing for rapid iteration and adaptation without compromising stability or introducing excessive complexity.

Modifiability is closely related to other quality attributes, such as maintainability, scalability, and testability. Maintainability refers to the ease with which a software system can be repaired or adapted after deployment. A highly modifiable system is also easier to maintain, as changes can be made quickly and with minimal disruption. Scalability refers to the ability of a system to handle increasing amounts of workload. A modifiable system can be easily scaled to meet changing demands, as new components can be added or existing components can be modified without requiring major changes to the overall architecture. Testability refers to the ease with which a software system can be tested. A modifiable system is also easier to test, as changes can be made in a controlled manner and the impact of those changes can be easily assessed.

Reusability-Driven Documentation and Testing in Distributed Systems

Building upon the principles of modifiability, a key aspect of modern software design is the ability to reuse components and services across different parts of the system. This reusability not only reduces development time and costs but also improves the overall consistency and maintainability of the software. However, achieving effective reusability requires clear and up-to-date documentation, as well as comprehensive integration tests that ensure that reused components function correctly in different contexts.

Traditional approaches to documentation and testing often struggle to keep pace with the rapid evolution of software systems, leading to inconsistencies and reduced reusability. To

address these challenges, this project explores the use of automated techniques to generate documentation and integration tests that are specifically tailored to promote reusability.

The core idea is to leverage Large Language Models (LLMs) to analyze the codebase and automatically generate:

- **API documentation that highlights reusable components**: This documentation would go beyond simply describing the API endpoints and parameters, and would also provide guidance on how to effectively reuse the components in different scenarios.
- **Integration tests that focus on verifying the reusability of services**: These tests would ensure that the services function correctly in different contexts and with different combinations of inputs, promoting confidence in their reusability.

By automating the generation of documentation and tests, this project aims to create a virtuous cycle where reusability is not only encouraged but also actively supported by the development process.

Patterns for Automated Documentation and Test Generation

To effectively automate the generation of API documentation and integration tests for reusable services, it is essential to identify and apply relevant design patterns. These patterns can serve as a blueprint for structuring the codebase in a way that facilitates automated analysis and generation of high-quality documentation and tests.

Instead of focusing on a broad "Microservices Pattern Language," this project will concentrate on specific patterns that directly support the automation goals. These include:

- **Document-Driven Development:** This approach emphasizes the creation of comprehensive API documentation as a primary step in the development process. By defining the API contracts and behavior upfront, it becomes easier to generate both documentation and tests automatically.
- **Testable Design:** This involves structuring the codebase in a way that makes it easy to write and execute automated tests. This can be achieved through techniques such as dependency injection, separation of concerns, and the use of well-defined interfaces.

By carefully selecting and applying these patterns, this project aims to create a codebase that is well-documented and easily testable.

Proposal

Motivation

Modern software development increasingly relies on distributed architectures composed of interconnected services. While this approach offers scalability and agility, it also introduces challenges related to documentation, testing, and reusability. This project is motivated by the need to address these challenges through AI, specifically Large Language Models (LLMs), to automate API documentation and integration test generation, promoting service reusability and reducing manual effort.

Recent exploration of reusability in microservices has resulted in a catalog of patterns and refactorings, but these lack tool support. This project bridges this gap by leveraging AI for Software Engineering, focusing on automated API documentation generation to ensure developers have the information they need to reuse services effectively, and automated integration test generation to verify reusability in different contexts, providing confidence in service reliability and compatibility.

Goals

This project aims to empower professionals in promoting and ensuring the reusability of services during the development and maintenance of Microservice Architecture (MSA)-based systems. An AI-powered system for automated API documentation and integration test generation will be developed and evaluated to support this goal. The expected outcome is a system that significantly improves the quality and coverage of API documentation, effectively verifies service reusability in different contexts, and ultimately enhances development efficiency, enabling professionals to build more reusable and maintainable MSA-based systems.

Methodology

This project will be conducted according to the following methodology:

Literature Review: This initial phase involves a comprehensive review of existing literature on API documentation, integration testing, AI for software engineering, and related topics. The focus will be on academic papers, articles, and publications on LLMs, RAG, OpenAPI (for both synchronous and asynchronous methods), and microservice patterns that promote reusability. The goal is to build a solid understanding of the current state-of-the-art, identifying relevant techniques, tools, best practices, and potential challenges, especially those related to providing tool support for conceptual reusability patterns. This review will also consider existing projects and research initiatives in the field of AI-powered API documentation and integration testing to identify potential synergies and avoid duplication of effort.

System Design and Architecture: The architecture of the AI-powered system will be designed based on the literature review. This phase will involve defining the components,

their interactions, and the data flow through the system. The design will evaluate integrating the system as a self-contained "black box" versus linking it to an existing development stack. Factors such as OpenAPI standards, potential IDE integrations (possibly through plugins), and automating code annotation to generate .yaml files will be taken into account. The expected result is an outline of the system architecture, indicating the chosen approach (black box vs. stack-linked) and the key components for automated documentation and test generation. This design will be iterative, with refinements based on feedback from stakeholders and preliminary implementation results.

Implementation and Prototyping: In this phase, the AI-powered system will be built, and a prototype will be created to demonstrate its functionality. The implementation will be based on the system architecture and design. The goal is to have a functional MVP that can automatically generate API documentation and integration tests for selected microservices, using LLMs and RAG. The implementation will follow agile development principles, with frequent iterations and continuous integration.

Evaluation and Refinement: Once the MVP is complete, its performance will be evaluated. This phase will involve assessing the MVP using a set of criteria (e.g., documentation coverage, test pass rate, and development time). The results will be used to identify strengths, weaknesses, and areas for improvement. The system will then be refined based on this feedback, addressing any limitations that are identified. The evaluation will involve both quantitative metrics (e.g., test coverage, documentation completeness) and qualitative feedback from users (e.g., ease of use, perceived usefulness).

Documentation and Thesis Writing: The final phase involves documenting the project, writing the thesis, and preparing presentation materials. This stage will involve synthesizing all the information from the previous phases, including the literature review, the system design, the implementation details, and the evaluation results. The outcome will be a well-documented thesis describing the project, its methodology, results, and contributions, including a discussion of the viability of RAG and potential integration with tools like GitHub Actions and Copilot. Slides and a poster will also be created to summarize the key findings. The thesis will also discuss the limitations of the project and potential directions for future research.

Expected Results

Upon completion of this project, the following results are expected:

1. **A functional AI-powered system for automated API documentation generation:** This system will automatically generate comprehensive and accurate API documentation from service code, adhering to industry standards such as OpenAPI. The system will leverage LLMs to analyze code and produce documentation that highlights key aspects of the API.

2. **A functional AI-powered system for automated integration test generation:** This system will automatically generate integration tests that

verify the reusability of services in different contexts. The system will leverage LLMs to analyze API contracts and generate test cases that simulate real-world interaction scenarios.

3. **An evaluation of the effectiveness of the AI-powered system:** This evaluation will assess the system's ability to generate accurate and comprehensive documentation and tests. The evaluation will involve both quantitative metrics and qualitative feedback from developers.

4. **A documented methodology for applying AI to automate API documentation and integration test generation:** This methodology will provide a guide for applying AI techniques to automate these tasks.

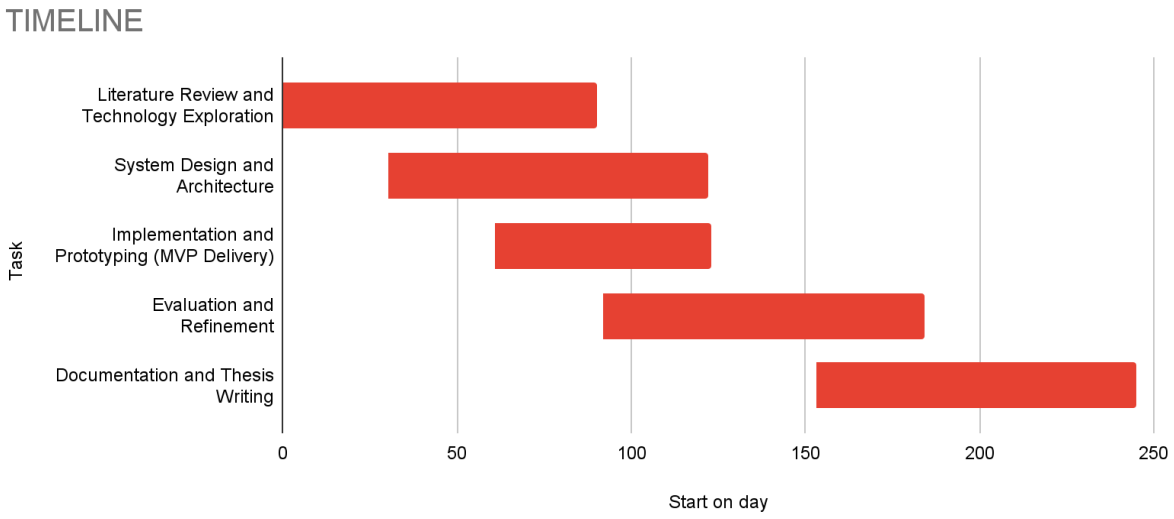
5. **A publicly available dataset of microservices code and associated documentation and tests:** This dataset will serve as a resource for researchers and developers working on AI for software engineering.

Workplan

Timeline

The project will be executed according to the following timeline:

- **April - June:** Literature Review and Technology Exploration
- **May - July:** System Design and Architecture
- **July - August:** Implementation and Prototyping (MVP Delivery)
- **August - October:** Evaluation and Refinement
- **October - December:** Documentation and Thesis Writing



References

Newman, Sam. *Building Microservices*. O'Reilly Media, 2015.