

REVERSE ENGINEERING MICROSERVICES

REVERSE ENGINEERING MICROSERVICES FOR ENHANCED
INSIGHTS

BY
MUHAMMAD WAQAR UL HASSAN AWAN, M.Eng.

A REPORT
SUBMITTED TO THE COMPUTING AND SOFTWARE
AND THE SCHOOL OF GRADUATE STUDIES
OF MCMASTER UNIVERSITY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTERS OF ENGINEERING

© Copyright by Muhammad Waqar Ul Hassan Awan, Nov 2024

All Rights Reserved

Masters of Engineering (2024)
(Computing and Software)

McMaster University
Hamilton, Ontario, Canada

TITLE: Reverse Engineering Microservices for Enhanced Insights

AUTHOR: Muhammad Waqar Ul Hassan Awan
M.Eng. Computing and Software,
McMaster University, Hamilton, Canada

SUPERVISOR: Dr. Sébastien Mosser

NUMBER OF PAGES: xiii, 27

Lay Abstract

A lay abstract of not more 150 words must be included explaining the key goals and contributions of the thesis in lay terms that is accessible to the general public.

Abstract

Abstract here (no more than 300 words)

Your Dedication
Optional second line

Acknowledgements

Acknowledgements go here.

Contents

Lay Abstract	iii
Abstract	iv
Acknowledgements	vi
Notation, Definitions, and Abbreviations	xi
Declaration of Academic Achievement	xiii
1 Introduction	1
2 Background	2
3 Vision and Technical Strategy	3
3.1 Vision	4
3.2 Technical Strategy	5
3.2.1 Framework Components	5
3.2.2 Probes	6
3.2.3 Single Source of Truth (SST)	14
3.2.4 Visualizer	16

3.3	Referencing	16
3.4	Figures	17
3.5	Tables	17
3.5.1	Long Tables	18
3.6	Equations	18
4	Implementation Report	20
5	Scenario Validations	21
6	Conclusion	22
A	Your Appendix	23
B	Long Tables	24

List of Figures

3.1	Software Maintenance Types	4
3.2	Probes and SST	15
3.3	Single Figure Environment Listed Title	17
3.4	A Multi-Figure Environment	19

List of Tables

3.1	A sample table	17
-----	--------------------------	----

Notation, Definitions, and Abbreviations

Notation

$A \leq B$ A is less than or equal to B

Definitions

Challenge With respect to video games, a challenge is a set of goals presented to the player that they are tasks with completing; challenges can test a variety of player skills, including accuracy, logical reasoning, and creative problem solving

Abbreviations

SST Single source of truth

UDS Unified data source

RE	Reverse engineering
SRE	Software reverse engineering
DSL	Domain-specific language

Declaration of Academic Achievement

The student will declare his/her research contribution and, as appropriate, those of colleagues or other contributors to the contents of the thesis.

Chapter 1

Introduction

Every thesis needs an introductory chapter

While you're here, you need to go into `definitions.tex` to set all the information needed for the front matter (e.g. title, author) and page header/footer.

You will also find the School of Graduate Studies' preparation guide (August 2021) for theses and reports. I would give it a quick read so you know what's expected.

Chapter 2

Background

Every thesis needs an Background chapter

While you're here, you need to go into `definitions.tex` to set all the information needed for the front matter (e.g. title, author) and page header/footer.

You will also find the School of Graduate Studies' preparation guide (August 2021) for theses and reports. I would give it a quick read so you know what's expected.

Chapter 3

Vision and Technical Strategy

The development of a large software system is a very critical process and there are a lot of parts that are involved in a project that needs to follow key software development paradigms so that it is easy to maintain. Its not just about writing code, its about creating a system that is bug free, easy to maintain, scalable and efficient.

Since an already built large software system is quite complex, so understanding this system requires certain strategies and tools. (Folmer et al., 2005) discusses about the useability issues in the software systems post-development and they require significant architectural changes. In order to deal with such architectural changes, understanding of the whole system is required and if the system is large enough, major resources are spent to fix even minor issues.

(Canfora and Cimitile, 2001) states that several surveys indicate that software maintenance consumes 60% to 80% of the total life cycle costs. Also the maintenance costs are largely due to enhancement (often 75-80%), rather than corrections. So a framework should be developed that could solve these issues by doing reverse engineering on the software systems and showing the importants aspects of the software

architecture in order to make it easy to understand and perform the required tasks.

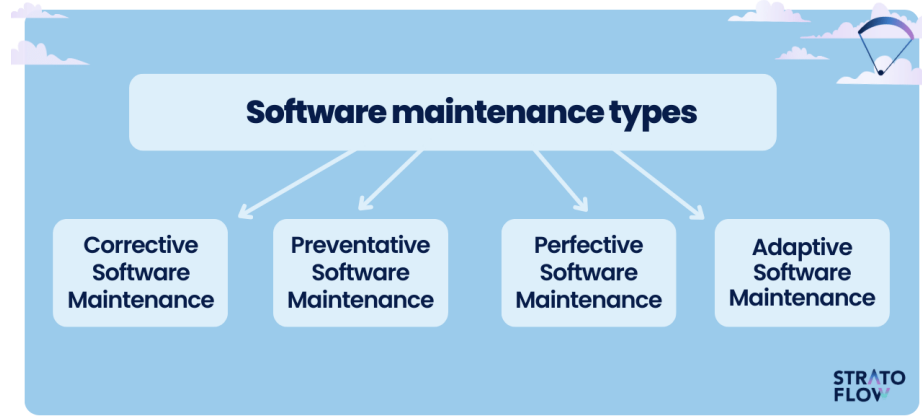


Figure 3.1: Software Maintenance Types (adapted from Stratoflow (2025))

3.1 Vision

Keeping in mind the above discussed issues, there is a need for an approach in which the software system could be analyzed, undergo processing and produce useful informations that can be used to provide enhanced insights about the project. The vision is to use a framework consisting of 3 components: Probes, Single Source of Truth (SST) and Visualizer. All three components will be discussed in detail in section 3.2.

In this report, we will discuss,implement and validate this framework by extracting static information from a project. In the future, this can be implemented/integrated with the CI/CD pipeline and provide dynamic information from the projects. The test project used in this report is Java spring framework based project called petclinic. Read more about this project from spring petclinic microservices github repository (Spring-Team, 2025).

Moreover, our approach will be mainly focused on the unified data source (UDS)

approach. (Statistics-Easily, 2025) states that Unified Data refers to the integration and consolidation of data from various sources into a single, cohesive framework. This approach allows organizations to streamline their data management processes, ensuring that all data is accessible and usable across different departments and applications. By unifying data, businesses can eliminate silos, reduce redundancy, and enhance the overall quality of their data analytics efforts. It means that consistent, up-to-date and valid data will be available using the UDS technique.

3.2 Technical Strategy

This section provides a detailed discussion of the components in the framework. It also includes an analysis of the probes used to prove the framework’s credibility, their use cases, and their benefits. After that, it discusses the unified data source (UDS) and, finally, the visualizer component.

3.2.1 Framework Components

The framework contains three main components.

- Probes
- Single Source of Truth (SST)
- Visualizer

3.2.2 Probes

The first component of the framework consists of **probes**. Probes represent distinct informational artifacts that are systematically extracted from software systems to provide insights and actionable data. In the context of this report, we have several use cases and extracted specific pieces of information that are detailed in the below subsections. These probes serve as the foundational elements for gathering critical data points that enable analysis and decision-making.

Looking forward, this concept can be expanded to more complex and targeted informational needs. By refining the scope and nature of the probes, we can tailor the probes according to our needs and to capture more refined data, addressing evolving requirements and insights that drive meaningful outcomes. This flexibility ensures that as the systems grow or change, the framework remains relevant and capable of producing deeper, more impactful information.

Given below are some of the probes that are implemented as part of this report in order to show the credibility of the presented framework. In later chapters, their implementation will be discussed at length followed by the validation of each scenario discussed here.

1. Authors Contributions

Author contribution probe extracts information about the developer who made the most recent changes to a method, list of all the developers who contributed to a method and highlights the developer with most contributions to a particular method. There are multiple use cases for this type of data.

Use Cases

- **Bug Fix:** When a bug is found in a method, the most recent author who updated the method would have the context of the recent changes that they have made and can help diagnose and resolve the issue faster. This will help the organization increase their productivity.
- **Reviews:** Looking at the most recent developer would give an idea of who to reach for documentation, clarification and review of the changes.
- **History Tracking:** Provides insights into who and why the person made the recent changes which can be used for documentation purposes.
- **Identify Developers Group:** Helps identify the group of developers who worked on a particular method, which in turn enables them to share and transfer knowledge in case the top contributor of the method is unavailable.
- **Ownership:** Tracks if the code has been worked on collaboratively or primarily by one individual.
- **Refactoring:** Ensures that all the contributors can be consulted and their insights are taken before performing any major refactoring.
- **Knowledge and Accountability:** Identifies the subject matter expert for particular methods and assigns ownership of the method to the top contributor for accountability and guidance.
- **Planning:** Helps management decide who should be involved in any major changes to the method based on past contributions.

Benefits

- **Improve Communication:** Facilitates and improve communication between team members by identifying relevant stakeholders.
- **Enhanced Planning:** Improves resource allocation, project planning and decision making for development tasks.
- **Transparency:** Promotes accountability among team members by making contribution history transparent.
- **Efficiency:** Increases team efficiency and issue resolving by involving right people for the job.

2. File Authors

This probe extracts information regarding the authors of a particular file. In software development, especially in large-scale microservices architecture, understanding the list of contributors to individual files provides significant value.

Use Cases

- **Ownership:** Identifying contributors of a particular file highlights the ownership and familiarity with the functionality of the file of the people associated with it.
- **Collaboration:** Identifying contributors reveals whether a file has been developed collaboratively or by a single individual.
- **Auditing and Compliance:** Contributor information is crucial for tracking accountability and compliance. Specially in open-source projects.

- **Maintenance:** Files with multiple contributors might lead to maintenance challenges. Having list of contributors help assign the tasks to the correct people.
- **Bug and Issue Assignment:** Assigning issues to contributors who are familiar with the relevant files improves resolution speed.

Benefits

- **Accountability and Quality:** Knowing who contributed to a file ensures accountability and encourages higher-quality contributions.
- **Team Collaboration:** Make team collaboration easier by identifying relevant stakeholders for discussions.
- **Reduces Risk:** Identifies files relying on single contributor which can help in workload distribution.

3. Author Relation

This probe extracts information regarding the strength of collaboration between two authors. Greater the collaboration, great will be the strength of bond between them. The “relation strength” metric quantifies the level of collaboration between pairs of authors based on shared contribution to files. In collaborative software development, especially in microservices architectures, understanding the strength of relation among authors can provide valuable insights into teamwork, communication and collaboration in the development process.

Use Cases

- **Collaboration Analysis:** Provides a quantitative measure of collaboration between two developers in a development team. Higher strength shows frequent joint contributions.
- **Planning and Resource Allocation:** Helps plan tasks and form teams for future projects by using existing strong collaboration bonds.
- **Knowledge Transfer:** New developers can use relation strength data to identify key collaborators within the team.
- **Shared Ownership:** Helps determine the shared ownership of the files, making it easier to assign maintenance tasks.
- **Technical Debt:** Low pairwise strengths and high number of collaborators of a file represents cohesive ownership, leading to technical debt.

Benefits

- **Improved team dynamics:** Improve team dynamics and encourages better interaction in areas with weaker team collaboration.
- **Fast Problem Solving:** Developers with high collaboration strength are likely to solve issues in shared files effectively.
- **Transparency:** Visualizes team interactions, increasing accountability and transparency.
- **Employee Evaluation:** Management can use the data to evaluate employees. Developers with higher collaboration strengths with multiple individuals shows employee value.

4. Microservices Endpoints

This probes extracts REST API endpoints from each service of a microservices project. Extracting such data provide valuable insights into how the application communicates and interacts with external systems or clients.

Use Cases

- **Documentation and Analysis:** Automatically extracts endpoints information to generate accurate and up-to-date documentation and perform analysis.
- **Application Functionality:** Provides overview of the functionality exposed by each service.
- **Testing:** Extracted endpoint data can be used to perform regression testing and prioritize test cases.
- **Security Audits:** Endpoint extraction aids in auditing APIs for potential security vulnerabilities.
- **Productivity:** Helps developers understand the microservices quicks and API offerings of each service.

Benefits

- **Debugging Issues:** Helps locating the faulty file and class, and helps developers quickly identify the method handling the request and resolve the issue.
- **Documentation:** Teams can use extracted endpoint data providing clients with API documentation.

- **Collaborations:** Facilitates communication between backend and frontend teams by providing endpoint insights.

5. Spring Beans

This probe extracts beans from java spring framework services. In Spring, the objects that form the backbone of your application and that are managed by the Spring IoC container are called beans. A bean is an object that is instantiated, assembled, and managed by a Spring IoC container. Otherwise, a bean is simply one of many objects in your application. Beans, and the dependencies among them, are reflected in the configuration metadata used by a container (Spring-Framework-Documentation, 2025).

Use Cases

- **Debugging and Maintenance:** Bean extraction provides a clear map of all components, aiding in debugging and maintenance activities.
- **Debugging and Maintenance:** Extracted bean data reveals the dependencies between components, helping teams understand coupling within the application.
- **Dynamic Bean Management:** Helps to enable dynamic and conditional bean registration and configuration.

Benefits

- **Improves Debugging:** Quickly locates missing or misconfigured beans, reducing the development downtime.
- **Security Improvement:** Detects potential security risks and stability issues.

6. Extracting Dependencies

This probe extracts dependencies from the POM (Project Object Model) files. A POM (Project Object Model) file is an XML file that serves as the fundamental building block of a Maven project. It contains information about the project and configuration details to manage dependencies, plugins, build settings, and more (Saurabh, 2023).

Use Cases

- **Dependency Management and Version Control:** Extracting dependencies helps track the versions of libraries and frameworks used across microservices.
- **Microservices Dependencies:** Analyze dependencies to understand the relationships between microservices and their shared libraries.
- **Performance Optimization:** Shows performance heavy libraries or those that introduce inefficiencies.

Benefits

- **Improves Performance:** Helps to identify, update or remove unused dependencies optimizing the performance and efficiency of the services.
- **Security Improvement:** Make it easy to identify those dependencies that are impacting the security of the services.
- **Maintainability:** Help keep all the dependencies up-to-date and easy to maintain.

3.2.3 Single Source of Truth (SST)

When data is being extracted from different sources, there are high chances of it becoming inconsistent and stale. If some sort of analysis is being done on the data, the input data needs to be up-to-date and reliable. In section 3.1 (Vision), we mentioned the visualizer component. It is supposed to run desired analysis on the data and provide visual updates. More on the visualizer in the upcoming subsection. So, if analysis is to be done on the input data, it needs to be a trusted, accessible, credible, reliable, and consistent.

Maintaining data quality is crucial for accurate insights and decision-making. Unified sources ensure that the input data remains clean and updated, regardless of its origin. Properly validated and processed data serves as the foundation for meaningful analysis and visualization.

(Müller et al., 2018) presents a scalable and extensible approach for software analysis and visualization by integrating data from diverse sources into a unified source. The paper discusses that having a unified data source provides a single access

point for querying diverse data, eliminating the need to manage multiple disconnected sources.

The single source of truth (SST) component in the introduced framework does the same job a unified data source. It is working as a data warehouse that takes all the information from different artifacts and process it. The tool used as a single source of truth in this project will be discussed in chapter 4 (Implementation Report) and the validation of the output provided will be discussed in chapter 5 (Scenario Validations).

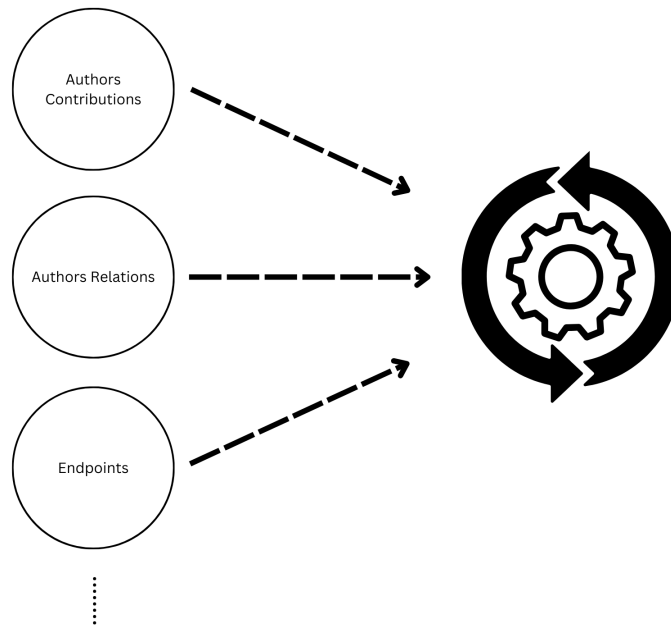


Figure 3.2: Probes and SST

Integration and Output

The probes will extract the required data from the microservices and feed them to the SST for further processing. SST will generate output which will be used by the

visualizer for further analysis and visualization.

Data Management

The output of the SST will be stored in a separate database to maintain and preserve historical records systematically. This approach ensures that the data is structured, organized, and easily retrievable from a centralized repository. By consolidating the data into a single repository, it not only simplifies access but also ensures a consistent format, which is critical for long-term usability and reliability.

Storing the data in a database enhances its integrity by eliminating inconsistencies and reducing complexities, such as duplicate entries. This structure allows analysts or analysis tools to focus on extracting meaningful insights without the additional work of cleaning or reorganizing raw data.

Additionally, a dedicated database supports scalability, meaning it can accommodate larger datasets as the system evolves. It also offers improved data management capabilities, such as access controls and backup solutions, ensuring the data remains protected and readily available for future use.

3.2.4 Visualizer

If you need to use quotes, type it “like this”.

3.3 Referencing

These are some sample references to GAMYGDALA (Popescu et al., 2014) from the `references.bib` file and state effects of cognition (Hudlicka, 2002) from the

references_another.bib file. These references are not in the same .bib file.

3.4 Figures

This is a single image figure (Figure 3.3):



Figure 3.3: This is a single figure environment

This is a multi-image figure with a top (Figure 3.4a) and bottom (Figure 3.4b) aligned subfigures:

3.5 Tables

Here is a sample table (Table 3.1):

A	\longleftrightarrow	B
C	\longleftrightarrow	D

Table 3.1: A sample table

3.5.1 Long Tables

A sample long table is shown in Appendix B.

3.6 Equations

Here is a sample equation (Equation 3.6.1):

$$y = mx + b \tag{3.6.1}$$



(a) Figure 1



(b) Figure 2

Figure 3.4: A Multi-Figure Environment

Chapter 4

Implementation Report

Chapter 5

Scenario Validations

Chapter 6

Conclusion

Every thesis also needs a concluding chapter

Appendix A

Your Appendix

Your appendix goes here.

Appendix B

Long Tables

This appendix demonstrates the use of a long table that spans multiple pages.

Col A	Col B	Col C	Col D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D

Continued on the next page

Continued from previous page

Col A	Col B	Col C	Col D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D
A	B	C	D

Bibliography

- Gerardo Canfora and Aniello Cimitile. 2001. Software Maintenance. *Handbook of Software Engineering and Knowledge Engineering* 1 (01 2001). https://doi.org/10.1142/9789812389718_0005
- Eelke Folmer, Jilles van Gorp, and Jan Bosch. 2005. Engineering Human Computer Interaction and Interactive Systems. In *Engineering Human Computer Interaction and Interactive Systems*, Rémi Bastide, Philippe Palanque, and Jörg Roth (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 38–58.
- Eva Hudlicka. 2002. This time with feeling: Integrated model of trait and state effects on cognition and behavior. *Applied Artificial Intelligence* 16, 7-8 (2002), 611–641.
- Richard Müller, Dirk Mahler, Michael Hunger, Jens Nerche, and Markus Harrer. 2018. Towards an Open Source Stack to Create a Unified Data Source for Software Analysis and Visualization. In *2018 IEEE Working Conference on Software Visualization (VISSOFT)*. 107–111. <https://doi.org/10.1109/VISSOFT.2018.00019>
- Adrian Popescu, Joost Broekens, and Maarten van Someren. 2014. GAMYGDALA: An emotion engine for games. *IEEE Transactions on Affective Computing* 5, 1 (2014), 32–44.

Saurabh. 2023. *Understanding the POM File in Spring: A Comprehensive Guide*. Dev.to. [https://dev.to/saurabhnative/understanding-the-pom-file-in-spring-a-comprehensive-guide-4gnl#:~:text=A%20POM%20\(Project%20Object%20Model,%2C%20build%20settings%2C%20and%20more](https://dev.to/saurabhnative/understanding-the-pom-file-in-spring-a-comprehensive-guide-4gnl#:~:text=A%20POM%20(Project%20Object%20Model,%2C%20build%20settings%2C%20and%20more.). Accessed: 2025-01-13.

Spring-Framework-Documentation. 2025. *Introduction to the Spring IoC Container and Beans*. Spring Framework Documentation Team. <https://docs.spring.io/spring-framework/reference/core/beans/introduction.html> Accessed: 2025-01-13.

Spring-Team. 2025. Spring Petclinic Microservices. <https://github.com/spring-petclinic/spring-petclinic-microservices>. Accessed: 2025-01-11.

Statistics-Easily. 2025. What is Unified Data? Understanding its Importance. https://statisticseasily.com/glossario/what-is-unified-data-understanding-its-importance/?utm_source=chatgpt.com#google_vignette Accessed: 2025-01-11.

Stratoflow. 2025. The Software Maintenance Process: What You Need to Know. <https://stratoflow.com/software-maintenance-process/> Accessed: 2025-01-11.