

**ENSURING MICROSERVICE ARCHITECTURAL
CONSISTENCY THROUGH AUTOMATED
DETECTION OF DRIFTS AND EROSIONS**

A.Z Jabir

229336D

Master of Science in Computer Science

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

April 2024

**ENSURING MICROSERVICE ARCHITECTURAL
CONSISTENCY THROUGH AUTOMATED
DETECTION OF DRIFTS AND EROSIONS**

A.Z Jabir

229336D

Thesis submitted in partial fulfillment of the requirements for the degree
Master of Science in Computer Science

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

April 2024

DECLARATION

I declare that this is my own work and this Thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature:

Date: 11.07.2024

The supervisor should certify the Thesis with the following declaration.

The above candidate has carried out research for the Master of Science in Computer Science Thesis under my supervision. I confirm that the declaration made above by the student is true and correct.

Name of Supervisor: Dr. Sunimal Rathnayake

Signature of the Supervisor:

Date: 12/07/2024

DEDICATION

It is with great honor and a sense of deep appreciation that I dedicate this project to my parents. Their unwavering love, support, and encouragement have been a source of strength and inspiration throughout my entire life and have made this achievement possible.

ACKNOWLEDGEMENT

First and foremost, I offer my sincerest gratitude to the Almighty for providing me with the strength, knowledge, and perseverance to complete this challenging yet rewarding journey.

I would also like to extend my sincere gratitude to my project supervisor, Dr. Sunimal Rathnayake, who guided me through every step of this journey with his expertise, patience, and support. His invaluable insights and encouragement have been instrumental in the successful completion of this research.

Finally, I would like to extend my heartfelt thanks to my colleagues, friends, and family who have supported me emotionally and provided me with the motivation to see this project through to its completion.

ABSTRACT

In the shift from monolithic to microservices architecture within the software development industry, the benefits of improved system resilience, agility, and decentralized data management are clear. Yet, this also introduces significant challenges in maintaining architectural consistency. The decentralized and heterogeneous nature of microservices imposes challenges in managing and maintaining the architecture integrity, which can lead to architectural degradation. This poses maintenance challenges and risks project success. The role of architectural integrity is crucial across the software lifecycle, affecting the development, deployment, and maintenance phases. Inconsistencies within the architecture can confuse the developers and make existing documentation outdated. Moreover, manual or semi-automated checks for architectural conformance are not only time-consuming but also prone to errors, highlighting the need for automated solutions. Recent studies have discovered the need for mechanisms that can proactively address architectural problems in microservices. The findings vouch that software architecture reconstruction is useful for addressing these architectural issues, with static analysis showing significant promise for the reconstruction of software architecture. However, there remains a notable gap in the utilization of language-neutral artifacts for automated reasoning and generating reports about the integrity of microservices architectures. This research presents a novel approach to ensure the consistency of microservice architecture by automating the detection of drifts and erosions. By adopting Model-Driven Engineering principles, which is widely in practice in the industry, our approach uses a textual technique to create a metamodel for each microservice and its communications, achieving technology neutrality. The process begins with a static analysis of the metamodel files and the intended architecture, followed by the construction of service dependency graphs to compare the intended and modified architectures. This systematic approach allows for the accurate identification of deviations, offering a preventive way to address architectural inconsistencies. To assess the effectiveness of this method, we evaluated it using the popular microservices demo application, OnlineBoutique. The findings demonstrate that our approach can precisely identify all microservices within the application and detect violations. These results suggest that our method is not only effective but also efficient in proactively identifying architectural drifts and erosions, contributing significantly to the maintenance of microservices architecture.

Keywords: Microservice Architecture, Software Architecture Degradation, Software Architecture Reconstruction, Architecture Conformance Checking , Model-Driven Engineering

TABLE OF CONTENTS

| | |
|---|------|
| Declaration of the Candidate & Supervisor | i |
| Dedication | ii |
| Acknowledgement | iii |
| Abstract | iv |
| Table of Contents | v |
| List of Figures | ix |
| List of Tables | xi |
| List of Abbreviations | xi |
| List of Appendices | xiii |
| 1 Introduction | 1 |
| 1.1 Chapter Overview | 1 |
| 1.2 Research Background | 1 |
| 1.2.1 Research Motivation | 1 |
| 1.2.2 Research Problem | 2 |
| 1.2.3 Research Questions | 2 |
| 1.2.4 Research Objectives | 2 |
| 1.2.5 Research Contribution | 2 |
| 2 Literature Review | 4 |
| 2.1 Chapter Overview | 4 |
| 2.2 Benefits of Microservices Architecture | 4 |
| 2.3 Challenges in Microservices Architecture | 5 |
| 2.3.1 Heterogeneous Nature | 5 |
| 2.3.2 Networking and Security | 5 |
| 2.3.3 Communication and Coordination | 6 |
| 2.3.4 Integration and Deployment | 6 |
| 2.4 Threats to Microservices Architecture Consistency | 6 |
| 2.5 Architecture Degradation in Microservices | 7 |

| | | |
|-------|--|----|
| 2.5.1 | Types of Architecture Degradation | 7 |
| 2.5.2 | Impact of Software Architecture Degradation | 7 |
| 2.6 | Approaches to Mitigate Architecture Degradation | 8 |
| 2.6.1 | Minimize | 8 |
| 2.6.2 | Prevent | 9 |
| 2.6.3 | Repair | 9 |
| 2.7 | Software Architecture Reconstruction | 10 |
| 2.7.1 | Static Analysis | 10 |
| 2.7.2 | Dynamic Analysis | 10 |
| 2.8 | Review of Existing Approaches to SAR in Microservices Architecture | 11 |
| 2.9 | Evaluation of the Existing Solutions | 17 |
| 2.9.1 | Expectations of the Practitioners | 17 |
| 2.9.2 | Key Metrics for Evaluation | 17 |
| 2.9.3 | Characteristics of the Preferred Solution | 18 |
| 2.9.4 | Analysis of the Existing Solutions | 19 |
| 2.9.5 | Summary of the Analysis | 21 |
| 2.10 | Summary of the Findings | 22 |
| 3 | Methodology | 23 |
| 3.1 | Chapter Overview | 23 |
| 3.2 | Solution Approach | 23 |
| 3.2.1 | Overview of the Approach | 23 |
| 3.2.2 | Introduction to Model-Driven Engineering | 23 |
| 3.2.3 | MDE in Meeting Solution Expectations | 24 |
| 3.2.4 | Metamodel Utilization | 25 |
| 3.2.5 | Static Analysis | 25 |
| 3.2.6 | Generation of Centralized-View | 25 |
| 3.2.7 | Identification of Drifts and Erosions | 25 |
| 3.2.8 | Integration with CI/CD Processes | 26 |
| 3.3 | Validation of the Solution | 26 |
| 3.3.1 | Evaluation Strategies Adopted in Previous Works | 26 |
| 3.3.2 | Evaluation Criteria Adopted for the Proposed Solution | 27 |

| | | |
|-------|---|----|
| 3.3.3 | Use Case for Experiments | 29 |
| 4 | Implementation | 30 |
| 4.1 | Chapter Overview | 30 |
| 4.2 | Choice of Architecture for Implementation | 30 |
| 4.2.1 | Cell-based Architecture | 30 |
| 4.2.2 | Model Abstractions | 30 |
| 4.3 | Design Goals | 32 |
| 4.4 | Overview of the Tool | 32 |
| 4.4.1 | Technology Usage | 33 |
| 4.4.2 | Inputs | 33 |
| 4.4.3 | Outputs | 33 |
| 4.5 | High-level Architecture | 34 |
| 4.5.1 | CLI | 34 |
| 4.5.2 | Static Analyzer | 34 |
| 4.5.3 | Degradation Detector | 35 |
| 4.5.4 | Reporter | 35 |
| 4.6 | Detecting Architecture Degradation | 36 |
| 4.6.1 | Flow Diagram | 36 |
| 4.6.2 | Sequence Diagram | 36 |
| 4.7 | CI/CD Integration | 37 |
| 5 | Evaluation | 38 |
| 5.1 | Chapter Overview | 38 |
| 5.2 | OnlineBoutique Microservices Application | 38 |
| 5.2.1 | Test Scenarios | 39 |
| 5.3 | TrainTicket Microservices Application | 41 |
| 5.3.1 | Test Scenarios | 42 |
| 5.4 | Results | 44 |
| 5.4.1 | Test Environment | 44 |
| 5.4.2 | Empirical Evaluation Results | 45 |
| 5.4.3 | Performance Evaluation Results | 45 |
| 5.4.4 | Analytical Evaluation Results | 47 |

| | | |
|-----|---|----|
| 6 | Discussion | 48 |
| 6.1 | Chapter Overview | 48 |
| 6.2 | Implications | 48 |
| 6.3 | Limitations | 49 |
| 7 | Conclusion and Future Work | 50 |
| 7.1 | Chapter Overview | 50 |
| 7.2 | Summary of the Contributions | 50 |
| 7.3 | Future Work | 50 |
| | References | 52 |
| | Appendix A Empirical Test Outputs - Online Boutique | 58 |
| | Appendix B Performance Test Outputs - Online Boutique | 62 |
| | Appendix C Test Outputs - TrainTicket | 65 |
| | Appendix D CI/CD Workflows | 70 |

LIST OF FIGURES

| Figure | Description | Page |
|---------------|---|-------------|
| Figure 4.1 | Cell-based Architecture Abstractions [1] | 31 |
| Figure 4.2 | High-level Architecture | 34 |
| Figure 4.3 | Flow Diagram | 36 |
| Figure 4.4 | Sequence Diagram | 37 |
| Figure 5.1 | Cell-based Architecture of OnlineBoutique Application | 39 |
| Figure 5.2 | Cell-based Architecture of TrainTicket Application | 42 |
| Figure A.1 | Output of test case 1 | 58 |
| Figure A.2 | Output of test case 2 | 58 |
| Figure A.3 | Output of test case 3 | 59 |
| Figure A.4 | Output of test case 4 | 59 |
| Figure A.5 | Output of test case 5 | 59 |
| Figure A.6 | Output of test case 6 | 60 |
| Figure A.7 | Output of test case 7 | 60 |
| Figure A.8 | Output of test case 8 | 61 |
| Figure B.1 | Output of performance test run 1 | 62 |
| Figure B.2 | Output of performance test run 2 | 62 |
| Figure B.3 | Output of performance test run 3 | 62 |
| Figure B.4 | Output of performance test run 4 | 63 |
| Figure B.5 | Output of performance test run 5 | 63 |
| Figure B.6 | Output of performance test run 6 | 63 |
| Figure B.7 | Output of performance test run 7 | 63 |
| Figure B.8 | Output of performance test run 8 | 64 |
| Figure B.9 | Output of performance test run 9 | 64 |
| Figure B.10 | Output of performance test run 10 | 64 |
| Figure C.1 | Output of attempt 1 | 65 |
| Figure C.2 | Output of attempt 2 | 65 |
| Figure C.3 | Output of attempt 3 | 66 |
| Figure C.4 | Output of attempt 4 | 66 |
| Figure C.5 | Output of attempt 5 | 67 |
| Figure C.6 | Output of attempt 6 | 67 |
| Figure C.7 | Output of attempt 7 | 68 |
| Figure C.8 | Output of attempt 8 | 68 |
| Figure C.9 | Output of attempt 9 | 69 |
| Figure C.10 | Output of attempt 10 | 69 |

| | | |
|------------|---------------------------------------|----|
| Figure D.1 | Solution Build Workflow | 70 |
| Figure D.2 | Inclusion of Conformance Check to PRs | 71 |
| Figure D.3 | Conformance Check Workflow | 72 |

LIST OF TABLES

| Table | Description | Page |
|--------------|---|-------------|
| Table 2.1 | Summary of the Analysis of Existing Solutions | 22 |
| Table 4.1 | Design Goals | 32 |
| Table 5.1 | OnlineBoutique Test Scenarios for Absence | 40 |
| Table 5.2 | OnlineBoutique Test Scenarios for Divergence | 41 |
| Table 5.3 | TrainTicket Test Scenarios for Absence | 43 |
| Table 5.4 | TrainTicket Test Scenarios for Divergence | 44 |
| Table 5.5 | Test Results for OnlineBoutique | 45 |
| Table 5.6 | Test Results for TrainTicket | 45 |
| Table 5.7 | Empirical Test Results | 45 |
| Table 5.8 | Performance Test Results with OnlineBoutique | 46 |
| Table 5.9 | Performance Test Results with TrainTicket | 46 |

LIST OF ABBREVIATIONS

| Abbreviation | Description |
|---------------------|--|
| ACC | Architecture Conformance Checking |
| CI/CD | Continuous Integration and Continuous Delivery |
| CLI | Command-Line Interface |
| FN | False Negatives |
| FP | False Positives |
| IDE | Integrated Development Environment |
| MDE | Model-Driven Engineering |
| SAR | Software Architecture Reconstruction |
| TP | True Positives |

LIST OF APPENDICES

| Appendix | Description | Page |
|-----------------|--|-------------|
| Appendix -A | Empirical Test Outputs - Online Boutique | 58 |
| Appendix -B | Performance Test Outputs - Online Boutique | 62 |
| Appendix -C | Test Outputs - TrainTicket | 65 |
| Appendix -D | CI/CD Workflows | 70 |