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

Improving Modifiability in Microservices Architectures: A reference implementation for a remote work platform in Software Startups

Thiago Guerrero

Work Proposal

mac 499 — Capstone Project

Supervisor:	Prof. Dr. Eduardo Martins Guerra
Co-supervisor:	MSc. João Francisco Lino Daniel

São Paulo
2023

Contents

Chapter 1

Introduction

Software architecture is an area that studies important concepts and practices to ensure the quality and success of a software system, hence improving its chances to succeed as a project. A good architecture allows you to have a broad view on a software system and its future evolution (Bass *et al.*, 1997). However, a software developed without a good architectural plan might be hard to keep up during its evolution. A poor architecture is a major cause of impediment for developers (Fowler, 2019).

Modifiability is a crucial quality attribute in software systems when it comes to be ready for future evolution, as it allows for more flexibility to use new frameworks, libraries, design patterns, programming languages, data sources, and so on (Bogner *et al.*, 2019). It is also closely related to other quality attributes such as maintainability, scalability, and testability. Though, designing for high modifiability is a huge challenge in Software Engineering, as it requires anticipating future changes and designing the system in a way that makes those changes easy to implement (Bogner *et al.*, 2019).

In this context, the present work proposal for the MAC0499 - Capstone project discipline aims to investigate the impact of modifications on microservices architecture and propose effective solutions to address some of the issues identified. This work will be conducted in the context of the Digi-dojo project being developed at the Free University of Bozen-Bolzano (unibz). The Digi-dojo project is a platform designed to support development teams in Software Startups that work remotely. The goal is to provide the interactive and creative environment of a startup for these teams. One of the focus of this project is the integration of existing Software Engineer tools with the platform, which has a microservices-based architecture (Wang *et al.*, 2022). As part of the scope of this work, one of the challenges will be to identify extension points on the platform to allow additional tools to be integrated in the future.

We will begin this document with a brief literature review of key concepts relevant to our work (Chapter 2). Following, we will provide an overview of our proposal (Chapter 3), outlining our motivation, goals, and methodology. Finally, we will present a table detailing the proposed timeline for the tasks and the expected results (Chapter 4).

Chapter 2

Literature Review

This chapter will cover some essential concepts in Software Architecture that are going to be important to understand this proposal.

2.1 Modifiability

The **Modifiability** is a quality attribute of the software architecture that measures the cost of making changes and reflects the ease with which a software system can accommodate modifications (Northrop, 2004). This attribute captures the cost in time and money of making a change, including the extent to which it affects other functions or quality attributes (Bass *et al.*, 1997).

A change can occur in any aspect of a system, such as internal functions, infrastructure, environment (dependencies, protocols, etc.), quality attributes (performance, reliability, etc.), capacity (Transactions Per Second (TPS) and limits), and new features/modification (Bass *et al.*, 1997). Thus, changes can be made by developers, installers, and even end users (Bass *et al.*, 1997).

The view on modifiability we will work with consists of two types of costs: the cost of preparing the system for changes and the cost of making changes. In other words, modifiability measures the ability of a system to easily incorporate a new solution as an alternative to existing options and to easily address new capabilities that the system did not previously support.

2.2 Microservices Architecture

The **Microservices Architecture (MSA)** is a software architectural style in which the system is composed by independently deployable, highly cohesive units – called microservices – that typically communicate via HTTP (Newman, 2015). Microservices are modeled around a specific business domain, encapsulating it via one or more well-defined network endpoints in order to expose it to other microservices or to external clients, making them a form of distributed system (Newman, 2020).

MSA offers several advantages, such as loosely coupled services, technology-agnosticism, autonomous decision-making, distributed responsibility and evolution between teams (as per Conway's Law), distributed blast radius, and independent risk assessments (Newman, 2020). For these reasons, it is often a suitable choice for teams, enabling them to manage complex systems and large engineering staff in smaller parts, improving scalability, robustness, modifiability, and other quality attributes.

Although the MSA has become very popular, it is not the best solution for all systems. It significantly increases the complexity of common tasks such as designing, infrastructure setup, and communication (mainly due to the network burden) (Newman, 2015). However, Those difficulties can be mitigated by using microservices patterns.

2.3 Microservices Pattern Language

A Pattern Language (PL) is an organized and coherent set of patterns, each of which describes a problem and the core of a solution that can be used in many scenarios (Alexander *et al.*, 1977). The **Microservices Pattern Language** is a specific type of PL which describes microservices' problems and their solutions. One of the well-known MSA PL embraced by the community is presented in Richardson, 2018, and is also available on *microservices.io*[1]. This pattern language encompasses a range of valuable patterns, some of which include:

- **API Gateway** provides an abstraction of the system APIs to their clients, making the microservices' system to look like it is a single component system.
- **Database per service** keeps each microservice's persistent data private and accessible only via its API. A microservice's transactions only involve its own database.
- **Event sourcing** persists the state of a business entity as a sequence of state-changing events. Whenever the state of a business entity changes, a new event is appended to the list of events. The application reconstructs an entity's current state by replaying the events.
- **Saga** implements each business transaction that spans multiple microservices as a saga. A saga is a sequence of local transactions. Each local transaction updates the database and publishes a message or event to trigger the next local transaction in the saga. If a local transaction fails because it violates a business rule, then the saga executes a series of compensating transactions that undo the changes that were made by the preceding local transactions.

[1] https://microservices.io/patterns/

Chapter 3

Proposal

3.1 Motivation

For large applications, a trending choice is Microservices Architecture (MSA). When confronting Modifiability and MSA, there is a convergence of factors. MSA emerged from a context of a wide range of techniques and technologies, and of limitations on the monolith (Newman, 2015). In a monolith, all responsibilities are bundled together into a single code-base, even when following a style like Layered Architecture, leading to difficulties in achieving the scale of modern software demands of large development teams and vast set of responsibilities (Richards, 2015). In other words, the nature of monolith limits Modifiability, leading professionals to adopt MSA to mitigate these limitations.

However, merely adopting an MSA-based architecture for a system does not immediately guarantee Modifiability. The ultimate goal with MSA is to be able to develop a new microservice and have it automatically incorporated to the rest of the system – the same goes for changes into an existing microservice. When that is not a reality, often a new feature in one microservice requires changes to others, resulting in additional work.

3.2 Goals

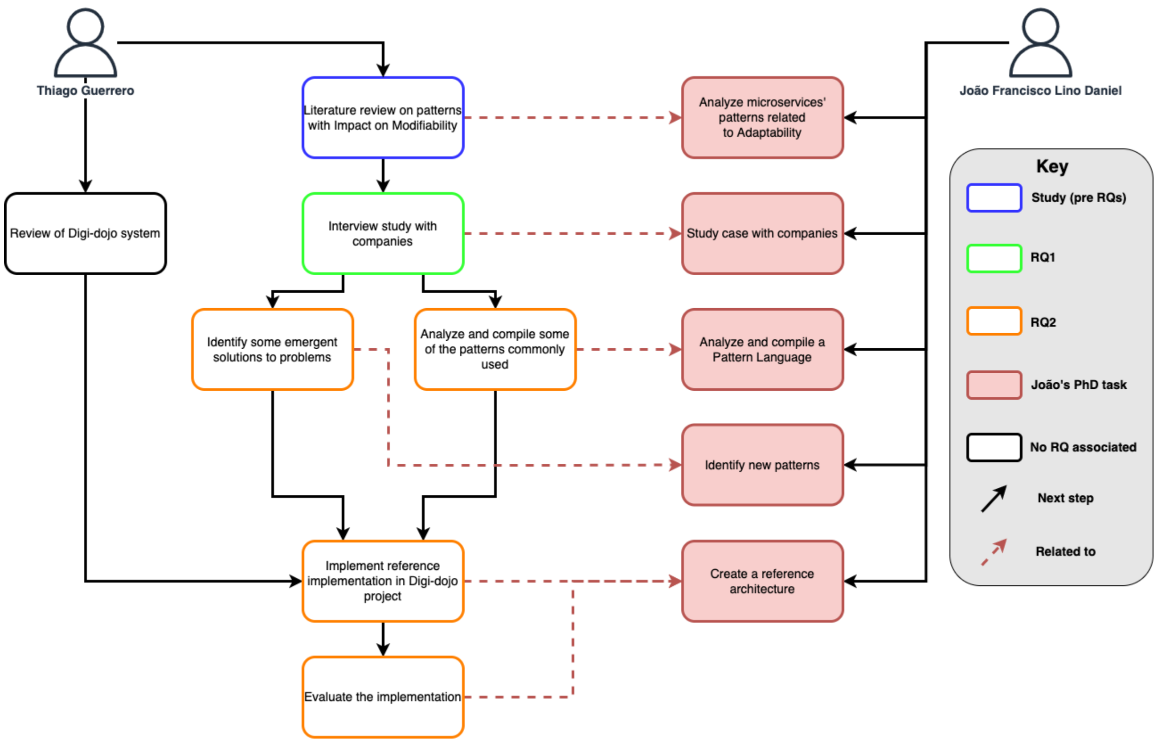
In this thesis, our objective is to investigate the impact of modifications in MSAs and propose effective solutions to address some of the issues identified. The work will be driven by the following set of Research Questions (RQ):

RQ1: *"How do the professionals deal with modifications in their MSA-based systems, in terms of practices, challenges, and solutions?"*

RQ2: *"How the currently adopted solutions address the challenges faced? And how the yet unhandled challenges can be addressed?"*

3.3 Methodology

An overview of the methodology that will be adopted in this work is illustrated in the figure 3.1.



Thiago Guerrero
João Francisco Lino Daniel
Literature review on patterns with Impact on Modifiability
Analyze microservices' patterns related to Adaptability
Review of Digi-dojo system
Interview study with companies
Study case with companies
Identify some emergent solutions to problems
Analyze and compile some of the patterns commonly used
Analyze and compile a Pattern Language
Identify new patterns
Implement reference implementation in Digi-dojo project
Create a reference architecture
Evaluate the implementation
Key
Study (pre RQs)
RQ1
RQ2
João's PhD task
No RQ associated
Next step
Related to

Figure 3.1: *Illustration of the steps (tasks) in the methodology of this work*

The first thing to notice is that the work will be related to João Francisco Lino Daniel's PhD work. In the next paragraphs, we will define the tasks and explain how they relate to João's work.

In the preliminary stage prior to defining the RQs, our aim is to do a literature review about a selection of patterns that enhance and promote modifiability at different levels of architecture. To achieve this goal, João is working on creating a collection of established patterns for his PhD work. So far, this work has selected 22 patterns, structured among six categories, and classified according to their influence on three dimensions: *Service Adaptability*, the ability of improving a service's behavior without changing its interface; *System Extensibility*, the capacity of the system to receive new microservice units; and *Microservice Extensibility*, the capacity of one single microservice unit to incorporate new services.

As the first step, we intend to conduct an interview study (RQ1), in particular with companies that use microservices daily. Our primary objective is to systematically identify and categorize the challenges faced by professionals in terms of Modifiability in MSA. Some of the topic we intend to understand include:

- What solutions do professionals adopt to address these challenges?

- Which challenges do not have a good solution?
- Which challenges are more frequent when designing?
- Why do professionals struggle to design for high modifiability?

Although this interview is listed in this proposal, it is not the main task of this work. In fact, the inputs collected from this study will be useful for the next tasks, and the results will be further explored in João's PhD work.

The following tasks involve two main efforts: compiling some of the patterns commonly used to address these challenges, and identifying some emergent solutions (RQ2). The first effort consists of establishing relationships between various patterns. During this analysis, it is possible to identify solutions not yet documented. This is going to be the initial effort of João's PhD task of compiling these patterns into a pattern language. In this way, the software engineering community can easily access, for instance, alternative solutions to the same problem — which empowers professionals to make better informed decisions.

In parallel with the preliminary stage and the interview study, we will make a review of the microservices being developed in the 76250A - Software Architecture discipline, being offered at Unibz in the second semester of 2022/2023[1] by Eduardo Martins Guerra, with João as teacher's assistant. In this discipline, second-year students are developing three distinct microservices – each one representing an integration with external software – as part of the Startup Digi-Dojo project. This code will be the starting point for the following tasks.

After completing the aforementioned tasks, we plan to do a reference implementation using the Digi-dojo project. Using the three microservices as the starting point, our objective is to implement some of the patterns compiled in the previous tasks and experiment some of the emergent solutions, focusing on making the project easier for future integrations. Finally, we will conduct an evaluation of the implementation in terms of how well the new architecture can handle modifications and the cost to evolve to this architecture. The result of this task will be used as input for João's work to create a reference architecture focused on adaptability to MSA-based architectures.

[1] The semester started on February 27th and ends on July 8th

Chapter 4

Work Plan

This chapter will propose a work plan based in the figure 3.1.

4.1 Timeline

The following table will present an expected timeline for the next months. Recapturing, the tasks are:

1. Literature review on patterns with impact on Modifiability.
2. Interview study with companies.
3. Analyze and compile some of the patterns commonly used.
4. Identify some emergent solutions to problems.
5. Review of Digi-dojo system.
6. Implement reference implementation in Digi-dojo project.
7. Evaluate the implementation.

Activities	April	May	June	July	August	September	October	November
1		✓	✓	✓				
2			✓	✓	✓	✓	✓	
3				✓	✓	✓	✓	✓
4						✓	✓	✓
5	✓	✓	✓	✓	✓			
6					✓	✓	✓	✓
7							✓	✓

Table 4.1: *Work plan's schedule*

4.2 Expected results

Upon completing all the tasks, we expect providing answers to the two Research Questions posed in the proposal chapter through the following set of results:

- A simple catalog of patterns with impact on Modifiability, mostly composed of commonly utilized solutions by professionals, along with some emergent patterns.
- A valuable contribution to the Digi-dojo project.
- An evaluation of the implemented patterns within the Digi-dojo project.

References

[Alexander *et al.* 1977] Christopher Alexander, Sara Ishikawa, and Murray Silverstein. *A Pattern Language: Towns, Buildings, Construction.* Oxford University Press, 1977. isbn: 0195019199 (cit. on p. 3).

[Bass *et al.* 1997] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice.* Third Edit. Addison-Wesley, 1997. isbn: 9780321711502 (cit. on pp. 1, 2).

[Bogner *et al.* 2019] Justus Bogner, Stefan Wagner, and Alfred Zimmermann. "Using architectural modifiability tactics to examine evolution qualities of service- and microservice-based systems". *SICS Software-Intensive Cyber-Physical Systems* 34 (2019). doi: 10.1007/s00450-019-00402-z (cit. on p. 1).

[Fowler 2019] Martin Fowler. *Software Architecture Guide.* 2019. url: https://martinfowler.com/architecture (cit. on p. 1).

[Newman 2015] Sam Newman. *Building Microservices - Design Fine-Grained Systems.* 2015. url: http://safaribooksonline.com (cit. on pp. 2–4).

[Newman 2020] Sam Newman. *Monolith to Microservices - Evolutionary Patterns to Transform Your Monolith.* 2020. url: http://safaribooksonline.com (cit. on pp. 2, 3).

[Northrop 2004] Linda Northrop. *Achieving Product Qualities Through Software Architecture Practices.* https://resources.sei.cmu.edu/asset_files/Presentation/2004_017_001_22862.pdf. 2004 (cit. on p. 2).

[Richards 2015] Mark Richards. *Software Architecture Patterns - Understanding Common Architecture Patterns and When to Use them.* 2015 (cit. on p. 4).

[Richardson 2018] Chris Richardson. *Microservices Patterns: With examples in Java.* First Edition. Manning, 2018. isbn: 9781617294549 (cit. on p. 3).

[Wang *et al.* 2022] Xiaofeng Wang *et al.* "Startup digi-dojo: a digital space supporting practice and research of startup remote work". *CEUR Workshop Proceedings* 3316 (2022). url: https://ceur-ws.org/ (cit. on p. 1).