Journal of Software Engineering Research and Development, 2026, 14:7, doi: 10.5753/jserd.2026.7762

Teaching Micro Frontends: Insights from Two Controlled Experiments on Guidelines and Anti-patterns

Nabson Silva [**Federal University of Amazonas** | ***nabson.paiva@icomp.ufam.edu.br***]
Eriky Rodrigues [**Federal University of Amazonas, FPFtech** | ***eriky.rodrigues@icomp.ufam.edu.br***]
Tayana Conte [**Federal University of Amazonas** | ***tayana@icomp.ufam.edu.br***]

Abstract

Context: Micro Frontend (MFE) is an architectural style that extends microservices principles to the frontend. Despite its growing adoption, misunderstandings about MFE foundations can create significant challenges during development. Preparing in-training software engineers to address these challenges and incorporating MFE into software architecture curricula is essential. **Goal**: We aim to address the gap in MFE education by presenting an experience report on teaching MFE in an undergraduate course. We compare two supporting materials to aid students in architectural decision-making: practitioner-provided guidelines and a catalog of MFE anti-patterns. Through two controlled experiments , we evaluate their effectiveness, with particular emphasis on understanding the role and benefits of using anti-patterns as a learning tool. **Method**: We taught MFE across five sessions and conducted a controlled experiment with two assessments, each one using one of the supporting materials. We compared them by analyzing differences in assessment scores and evaluated whether the catalog improved students' perceived learning. Additionally, we investigated how students used the catalog by applying the Technology Acceptance Model and collecting qualitative feedback regarding its use. Finally, we extend our previous work by introducing a new version of the catalog and comparing it with the original through a second controlled experiment. **Results**: Both supporting materials are equally helpful for solving MFE architectural problems. Students reported an increased perception of learning after engaging with the catalog. Their feedback indicated that the catalog was used to identify problems and solutions, promote efficient search for issues, and reinforce MFE knowledge. The results of the second experiment show no statistically significant differences between the versions. However, qualitative feedback indicates that preferences depend on individual reading styles and information needs. It also revealed additional ways students use the catalog to learn about MFE, such as using anti-patterns as a third learning phase. **Conclusion**: This paper provides insights into teaching MFE, introduces two supporting instructional materials, and highlights the value of anti-pattern catalogs for both education and practice. Our findings show that the catalog can support learning while also helping developers analyze architectural problems and make more informed decisions.

Keywords: *Micro frontends, Experience Report, Controlled Experiment, Empirical Software Engineering, Anti-patterns.*

1 Introduction

Micro Frontend (MFE) is an architectural style that extends the principles of microservices to the frontend, breaking down a complex frontend application into smaller, manageable, and independently deployable slices (Mezzalira, 2021a; Peltonen et al., 2021). This approach facilitates independent testing, development, and deployment of frontend components, enabling teams to work autonomously and reducing the development time for new features (Geers, 2020). However, some issues faced when adopting MFEs are increased payload size, UX consistency, complex monitoring and debugging, state management, and duplicated code (Peltonen et al., 2021). Many companies, such as SAP, Springer, Zalando, NewRelic, Ikea, Starbucks, and DAZN, have successfully implemented the MFE architecture (Taibi and Mezzalira, 2022), showcasing its potential in diverse domains.

Despite widespread adoption in the industry, MFE has yet to be incorporated into the software architecture course curriculum. This gap reflects the scarcity of academic research on MFE education, contrasting with the abundance of experience reports and case studies detailing its implementation (Antunes et al., 2024; Capdepon et al., 2023; Kaushik et al., 2024; Perlin et al., 2023; Männistö et al., 2023; Pölöskei and Bub, 2021; Moraes et al., 2024). Without formal educational resources, practitioners may encounter challenges in effectively implementing the architecture, potentially leading to suboptimal outcomes that hinder the realization of its full benefits. As educators, it is important to teach MFE and understand how to teach them effectively. It includes investigating which instructional strategies foster deeper learning through research-validated instruments or practice-oriented materials. Since MFE is still underrepresented in software architecture education, identifying practical and pedagogically sound ways to introduce it into the curriculum is essential to bridge the gap between academic training and current industry demands.

This paper addresses the gap in MFE education by sharing our experience teaching Micro Frontends in an undergraduate computer science course. We designed and evaluated two supporting materials to help students learn about MFE and make architectural decisions: a presentation with practitioner-provided guidelines and a web-based anti-patterns catalog we introduced in a previous study (Silva et al., 2025a). Both materials expose students to real-world scenarios and show the types of problems developers face when working with MFE. We delivered five sessions, including a hands-on lab, and assigned two assessments focused on maintenance and evolution tasks, each using a different instructional material. By presenting the structure of the ses-

sions, the assessment activities, and the students' results and feedback, we offer practical insights to help educators introduce MFE into software architecture courses and explore practical ways to teach it.

Beyond reporting our experience on teaching MFE, this paper presents a controlled experiment designed to compare the effectiveness of two supporting materials used by students. We also examine whether our catalog of MFE anti-patterns can enhance students' perceived learning. Unlike traditional materials, the catalog documents common architectural mistakes observed in practice and their corresponding solutions. Exposing students to these real-world problems and actionable guidance can foster a more practice-oriented understanding of MFE. Furthermore, we investigate how students used the catalog during assessments, aiming to understand its potential as a support tool for learning and architectural decision-making in MFE development.

Results show that both approaches equally supported students in learning MFE and solving maintenance problems in MFE architectures. We observed that the students' perceived learning increased after engaging with the catalog. Qualitative feedback revealed that the catalog was actively used to identify problems and solutions, support efficient search for issues, and reinforce MFE knowledge. These findings indicate that the catalog is a valuable learning and support tool by presenting real-world architectural challenges in a structured, accessible, and instructionally effective format. At the same time, the practitioner-provided guidelines also proved effective, offering a more didactic and introductory path to understanding MFE. Together, these results highlight two valuable educational resources that can support different learning needs in software architecture courses.

This paper is an extended version of Silva et al. (2025b). We present a new version of the MFE anti-patterns catalog, with 15 new anti-patterns and new fields to describe each one based on the first experiment feedback. To assess wether the new version is more useful than the previous one, we conducted another controlled experiment with undergraduate students to compare the two versions. We analyzed the results of the new controlled experiment in comparison to the previous one, which allowed us to expand and deep our previous analysis and learn new lessons on teaching MFE. Our results show that both catalog versions equally support students in solving MFE architectural problems, but qualitative feedback indicates that preferences depend on individual reading styles and information needs. We identify additional pedagogical roles for anti-pattern catalogs, including their use as a dedicated learning phase after introducing architectural concepts and their role in reinforcing the importance of careful architectural decision-making.

This work contributes to software engineering education by presenting a concrete teaching case that offers practical evidence and insights into teaching MFE effectively. We provide educators with a tested methodology and two supporting materials to help students' learning of MFE. Educators can adopt the detailed lecture structure, hands-on lab sessions, and assessments based on realistic MFE scenarios to enrich their software architecture courses. Beyond instructional design, our findings highlight the importance of using real-world examples and accessible tools to support students' learning. In particular, student feedback underscores the value of the anti-patterns catalog in helping learners identify common pitfalls and reason about architectural decisions. These contributions empower educators to expand curricula with practical, research-informed strategies for teaching modern architectural styles such as MFE.

2 Background

This section explores the fundamental principles of MFE and anti-patterns and discusses related work.

2.1 Micro Frontends

The term "Micro Frontend" was first coined by ThoughtWorks (2016) in 2016 as an architectural style inspired by microservices architecture. MFE decomposes a monolithic frontend application into smaller parts that can be developed, deployed, and updated independently, promoting greater flexibility and maintainability (Mezzalira, 2021a; Peltonen et al., 2021). Geers (2020) considers MFE as an organizational approach since the application is divided into vertical slices built from the database to the user interface and owned by dedicated teams based on the business sub-domains.

Implementing MFE offers several benefits, including support for different technologies, autonomous cross-functional teams, improved testability, and independence in development, deployment, and management (ThoughtWorks, 2016; Taibi and Mezzalira, 2022; Mezzalira, 2021a). However, this architectural style introduces additional complexity, making it challenging to monitor and debug, maintain a consistent User Experience (UX) across all MFEs, and manage shared dependencies without increasing payload size, among other issues (Geers, 2020; Peltonen et al., 2021).

To integrate MFEs and deliver a unified application, developers must implement Frontend Integration, which comprises Composition, Communication, and Routing. Composition involves requesting fragments—reusable components exported from one MFE to be used by another—and placing them in the appropriate slots on a screen (Geers, 2020). There are three approaches to composing MFEs: Server-side, Edge-side, and Client-side (Taibi and Mezzalira, 2022; Peltonen et al., 2021; Geers, 2020; Moraes et al., 2024). Communication defines how the screen and fragments interact to deliver a seamless user experience (Taibi and Mezzalira, 2022). According to Geers (Geers, 2020), communication can occur in three primary forms: Parent to Fragment, Fragment to Parent, and Fragment to Fragment. Finally, Routing handles navigation between views, typically implemented using hyperlinks (Taibi and Mezzalira, 2022).

2.2 Related Work

Previous studies have identified several challenges in teaching software architecture. According to Galster and Angelov (2016), students accustomed to seeking optimal programming solutions often find the "wicked" nature of architectural decision-making frustrating. The same study notes that

small classroom examples often fail to convey the importance of architectural decisions, while the scarcity of realistic examples limits students' practical learning opportunities. Kazman et al. (2023) argue that undergraduate students often lack development experience and tend to think like programmers, which makes it difficult for them to understand high-level architecture and abstraction, as they focus on implementation details rather than high-level design. Lago and Van Vliet (2005) highlight that the absence of stakeholders to provide clear business rules during the software architecture design process poses a significant challenge. To improve software architecture courses, Mannisto et al. (2008) state that they should emphasize using existing systems for maintenance and evolution tasks rather than solely designing architectures from scratch. This approach is more aligned with industry practices, where software architects commonly work with established systems.

Oliveira et al. (2022) systematic mapping results show that many software architecture courses focus mainly on conceptual foundations, such as quality attributes, architectural views, and general design principles, while giving limited attention to newer architectural styles and industry practices. Galster and Angelov (2016) highlight the importance of teaching architecture through concrete examples rather than relying only on abstract explanations. In this sense, do Nascimento Oliveira et al. (2025) reports four experiences on teaching software architecture using real-world cases. They emphasize the need to provide contextual support for students to understand the cases, adapt technical language to improve comprehension, tailor cases to avoid losing instructional goals, and progressively introduce more complex cases to evaluate students' capability to address new knowledge.

To the best of our knowledge, no papers specifically discuss MFE education, and only a few report on experiences teaching microservices. Bærbak Christensen (2022) describe the curriculum of an undergraduate course centered on DevOps and microservices, with a stronger emphasis on DevOps. They provide teaching guidelines that include focusing solely on architectural migration without adding new features to microservices to avoid overcomplication, using a technology stack familiar to students, and defining a monolith with clear domains and boundaries. Similarly, Lange et al. (2019) report on a microservices course conducted in collaboration with industry, where students attended lectures before migrating a monolith to a microservices architecture. Cordeiro et al. (2019) propose an approach for teaching microservices involving lectures delivered by researchers and industry professionals. In their approach, students are organized into teams and tasked with developing different domains of a system, simulating real-world collaboration and distributed development. Overall, these microservices courses emphasize implementing new architectures migrated from monoliths but lack focus on exercising architecture design itself.

3 Developed Supporting Materials

Due to time constraints, we could not let students implement an MFE architecture. Instead, we developed two supporting materials to help students learn about common problems and solutions in real MFE architectures: (1) a catalog of MFE anti-patterns and (2) a presentation with practitioner-provided guidelines.

3.1 Catalog of Micro Frontends Anti-patterns

Patterns provide reusable solutions to common problems, outlining the benefits of these solutions and the challenges architects must overcome to implement them successfully Richardson (2018). In contrast, anti-patterns identify a problem and present two potential solutions: one that is commonly adopted but ineffective, and another that is more effective Brown et al. (1998). Understanding both patterns and anti-patterns equips developers with the knowledge to design robust architectures while avoiding the recurring issues highlighted by these concepts.

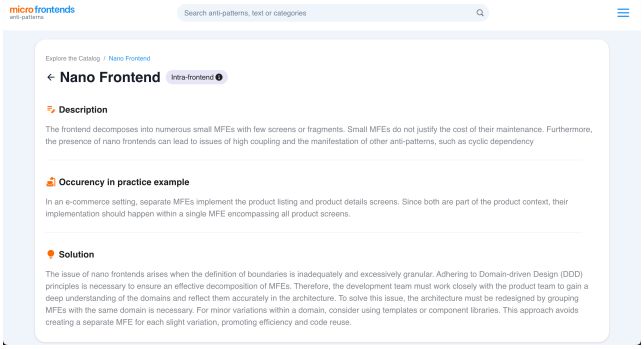

Figure 1. Nano Frontend anti-pattern description from the initial version of the catalog's web application.

In our previous research Silva et al. (2025a), we developed a catalog of 12 MFE anti-patterns and validated it through a survey with industry practitioners. Our catalog is publicly available through a web application to promote its use and facilitate collaboration among developers. Each anti-pattern is defined by a name, problem, solution, category, and example. Figure 1 presents the screen with the Nano Frontend anti-pattern description. However, this version is no longer available since we evolved the catalog (see Section 7).

3.2 Practitioner-Provided Guidelines

The practitioner-provided guidelines consists of a presentation in which we showcased the MFE architectures published by Antunes et al. (2024), Moraes et al. (2024), and Silva (2024). For each architecture, we explained how the MFEs were composed and added questions to help students identify potential problems and propose improvements. We also added a potential solution to each problem. Figure 2 presents a slide containing a problem and its potential solution of the architecture from Silva (2024). To conclude, we presented a summary of MFE guidelines published by Taibi and Mezzalira (2022), Aplyca (2024), Kofler (2020), Anks (2023), and Shukla (2023), consolidating recommended practices and solutions.

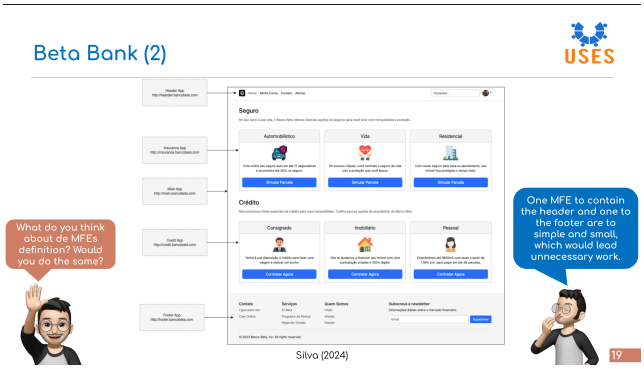

Figure 2. Slide from the practitioner-provided guidelines presentation with an example of architecture and the question we asked to students and its answer.

4 Teaching Approach

This section describes the procedures for teaching MFE to undergraduate Computer Science students at the Federal University of Amazonas (UFAM). This teaching process was conducted before the experiments to ensure that students had the necessary background on MFEs before participating in the studies. Before each study, we delivered theoretical and practical sessions to introduce MFEs to the undergraduate students taking the course. We delivered the same sessions and applied the same assessments in both studies.

4.1 Learning Objectives

Students were in the sixth semester of their undergraduate degree in Computer Science and were enrolled in the Systems Analysis and Design course, which focuses on topics related to software modeling and architecture. The course curriculum covered system design using UML diagrams, architectural styles, and design patterns. We delivered the MFE sessions after the topics on UML diagrams and software architecture styles to achieve the following learning objectives for MFEs:

1. Understand the benefits, challenges, and appropriate contexts for implementing MFE.
2. Learn best practices for designing and maintaining MFE architectures.
3. Develop the ability to assess and evaluate MFE architectures.

4.2 Sessions Content

We delivered 4 theoretical lecture sessions and 1 laboratory session. Table 1 presents the sessions and its content. Before the first class, students already had knowledge about architectural styles like Layered Architectures, Data-centered architectures, and Pipers & Filters (Valente, 2020), and about architecture visualization with C4 Model (Brown, 2023).

Since MFE is based on microservices, the first lecture session aims to introduce the definition, benefits, and challenges of microservices. The lecture begins by detailing the monolithic architectural style and the issues that motivated the creation of microservices (Newman, 2021). Then, we presented the concept of microservices and their key principles. To illustrate some architectural examples, we discussed

Table 1. MFE sessions' content.

#	Session	Description
1	**Microservices**	Monoliths, microservices, and patterns for microservices.
2	**Micro Frontends**	Definition, benefits, challenges, and frontend integration.
3	**Micro Frontends Hands-on**	Lab session where students implemented routing, composition, and communication in a web e-commerce application.
4	**Examples of Micro Frontends Architectures**	Public MFE examples and guidelines from experience reports, case studies and blogs.
5	**Micro Frontends Anti-patterns**	Definition of anti-patterns and an explanation of the 12 MFE anti-patterns.

the following patterns: Remote Procedure Invocation, Asynchronous Messaging, API Gateway, Backend for Frontend (BFF), and API Composition (Richardson, 2018).

In the second lecture session, we introduced the concept, benefits, and challenges of MFE primarily based on Geers (2020) and Peltonen et al. (2021). We also delved into implementing frontend integration through composition, communication, and routing. Finally, we presented an MFE architecture we developed, whose source code students would access during the hands-on session in the following session. The primary goal of this lecture was to provide the theoretical foundation of MFE.

During the third session, we conducted a lab session where students could engage with implementing an e-commerce application built using 3 MFEs [1], which we presented in the previous session. In the lab session, we explained how the application uses the Single-SPA framework (Single-spa, 2016) to compose the MFEs and then assigned three exercises focused on routing, composition, and communication. Our goal in this session was to make students see how MFE works in practice since its concepts may be confusing.

In the fourth and fifth sessions, we presented the two proposed supporting materials: practitioner-provided guidelines (Taibi and Mezzalira, 2022; Aplyca, 2024; Kofler, 2020; Anks, 2023; Shukla, 2023) and a catalog of 12 MFE anti-patterns (Silva et al., 2025a), respectively. Our goal was to introduce best practices for developing MFE architectures and to examine published examples, encouraging discussion about architectural decisions and how they could be improved. This approach aimed to enhance students' ability to assess MFE architectures critically. The final session was essential for helping students recognize common problems and effective solutions by exploring MFE anti-patterns.

4.3 Assessments

We conducted two comparable assessments related to MFE, each containing similar questions focused on analyzing two distinct architectures. The questions simulated tasks a new

[1] https://github.com/nabsonp/mfe-hands-on

developer might face when joining a team working with MFE architectures. The two systems—an e-commerce platform and a digital bank-were inspired by real-world MFE implementations. We selected these domains because students were already familiar with them, reducing the risk of confusion due to unfamiliar business contexts. Each assessment consists of two parts:

1. **Architecture Analysis**: we asked students to evaluate the proposed MFE architecture and indicate whether they would design it differently, providing justifications for their decisions.
2. **Maintenance and Evolution**: eight questions that required students to develop a new feature, understand and resolve a bug in the application, or refactor a part of the architecture.

To ensure the realism of the assessments and the proposed architectures, we asked two experienced MFE professionals to complete the assessments and provide feedback on whether the problems reflected real-world scenarios and whether the architectures resembled those commonly seen in the industry. We chose not to ask students to design a MFE architecture from scratch, as, in practice, they are more likely to maintain and evolve an existing architecture than creating one from scratch (Galster and Angelov, 2016; Kazman et al., 2023; Mannisto et al., 2008). We ensured that every question could be answered using either the anti-patterns catalog or the practitioner-provided guidelines, allowing for a fair comparison between the two supporting materials.

5 First Experiment Design

To address the first study's research questions (Section 5.1), we designed a controlled experiment following the guidelines proposed by Wohlin et al. (2012). The following subsections provide a detailed description of each aspect of the first experiment.

5.1 Goal and Research Questions

The experiment's goal is to explore effective teaching strategies for MFE by comparing the MFE anti-patterns catalog with practitioner-provided guidelines as supporting materials. Additionally, we aim to analyze whether the MFE anti-patterns catalog enhance students perceived learning and how it can be used during MFE maintenance. Therefore, we aim to answer the following set (A) of Research Questions (RQ):

RQ-A1

Which supporting material–an MFE anti-patterns catalog or practitioner-provided guidelines–leads to higher student assessment scores?

To address RQ-A1 , we compared the mean scores from the two MFE assessments. In the first assessment, students consulted practitioner-provided guidelines; in the second, they used the MFE anti-patterns catalog.

RQ-A2

Does the catalog of MFE anti-patterns enhance students' perceived learning about MFE?

For RQ-A2 , we asked students to rate their perceived learning about MFE before and after engaging with the catalog and then compared the mean of the two sets of responses.

RQ-A3

How do students use the MFE anti-patterns catalog, and do they intend to adopt it when solving MFE challenges?

For RQ-A3 , we evaluated the catalog's utility, ease of use, and students' intention to use it in the future by applying the original Technology Acceptance Model (TAM) (Venkatesh and Bala, 2008). In addition to the TAM constructs, we included four statements to assess how the catalog supported learning about MFE. We also collected qualitative feedback on how students used the catalog and analyzed it through Grounded Theory procedures (Corbin and Strauss, 2014).

5.2 Planning

We planned the experiment according to Wohlin et al. (2012).

5.2.1 Context Selection

We conducted the experiment with undergraduate students learning about MFE for the first time during the training sessions, without prior experience in this architectural style. This setup simulates junior developers entering a company and needing to work with MFE architectures. Since no published MFE architecture specifications described complete applications—including the MFEs implemented, their screens and fragments, and their communication and composition strategies—we developed two MFE applications for students to analyze during the experiment.

5.2.2 Variable Selection

The independent variable is the supporting material consulted during the assessments, with two treatments: (1) the catalog of MFE anti-patterns and (2) the practitioner-provided guidelines. The dependent variables are the students' assessment scores and perceived learning scores. Assessment scores are real values ranging from 0 to 10. To measure perceived learning, we asked students to self-assess their learning on MFE using real values ranging from 0 to 5, enabling statistical comparison between samples (Wohlin et al., 2012) and evaluating whether the catalog positively influenced students' learning perception, following the approach of Meireles et al. (2024).

5.2.3 Hypothesis Formulation

We formulated two hypotheses, each corresponding to RQ-A1 and RQ-A2 . The null hypothesis H_0, related to RQ-A1 , states that there is no significant difference in students' mean

assessment scores when supported by practitioner-provided guidelines compared to when using the anti-patterns catalog. The second null hypothesis, H_0', related to RQ-A2 , states that there is no significant difference in students' perceived learning before and after interacting with the MFE anti-patterns catalog. As RQ-A3 is addressed through qualitative methods, we did not define a hypothesis for it.

5.2.4 Participants Selection

Participant selection was based on convenience sampling (Wohlin et al., 2012), targeting undergraduate Computer Science students enrolled in the Systems Analysis and Design course at UFAM. Participation in the study was voluntary, and only students who signed the consent form and attended at least all but one session were included in the experiment.

5.2.5 Study Design

Students completed the two assessments described in Section 4.3. We designed two MFE architecture specifications (Objects) to support the assessments: Object 1 represented an e-commerce web application, while Object 2 represented a mobile application. To compare the proposed supporting materials (treatments), we employed a crossover design (Vegas et al., 2015) applied to the specification Objects, rather than to the treatments. This decision was necessary because the treatments required prior instruction and could not be delivered independently for each group without risking contamination threat validity (Wohlin et al., 2012). Since students continued attending shared sessions over time, separating them into different groups would not prevent cross-influence. Therefore, we alternated only the architecture objects analyzed in each assessment to maintain comparability, minimizing bias.

The students were divided into two groups, Group A and Group B. Since none of the students had prior experience with MFE, we balanced the groups based on software development experience type (backend, frontend, or full-stack) and duration, if any. During the first assessment, both groups consulted the presentation from the fourth session, which included practitioner-provided guidelines and MFE examples. Group A completed the first assessment based on Object 1, while Group B completed it based on Object 2. For the second assessment, both groups accessed a web application containing the MFE anti-patterns catalog, with the Objects reversed: Group A analyzed Object 2, and Group B analyzed Object 1. Figure 3 summarizes the study design.

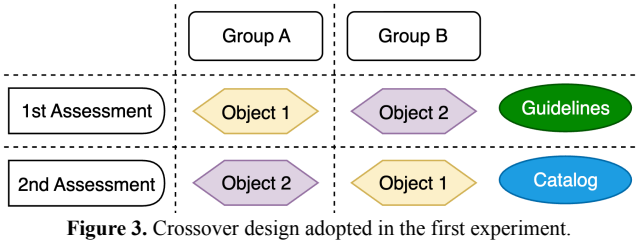

Figure 3. Crossover design adopted in the first experiment.

5.2.6 Instrumentation

The instruments for this experiment include a set of forms, training session documents, architecture descriptions, and an activity script that must be followed during the execution phase. We describe each instrument in detail as follows:

1. **Theoretical training**: Presentations delivered during lecture sessions and the code and script presented during the laboratory session.
2. **Consent form**: form that outlines the purpose of the experiment and details how it will be conducted. We emphasize that participation is voluntary, allowing participants to withdraw from the experiment at any time without affecting their scores on the assessments. Furthermore, we explain that we will use the collected data for quantitative and qualitative analysis and may include it in scientific publications. Participants signed the consent form before the experiment.
3. **Characterization form**: a set of questions designed to gather information about the participants' professional experience as software developers and their familiarity with MFE architectures.
4. **Objects**: description of two MFE architectures that students evaluated during the experiment. The objects include a brief description of the software and its functionalities, images showcasing its screens, and a comprehensive list of MFEs detailing their context, screens, and fragments.
5. **Assessment Forms**: Two forms, each containing the questions from them assessments described at Section 4.3.
6. **Feedback form**: This form included the TAM constructs, the learning-related statements, and the open-ended questions described in Section 5.4.

5.3 Execution

To ensure the realism of the specifications, we conducted a pilot study to ask two experienced MFE professionals to review the applications. They confirmed that the architectures and the problems presented in the assessments reflected real-world scenarios.

Figure 4 presents the sequence of steps followed during the experiment execution. Before the execution, we delivered the four primary lecture sessions described in Subsection 4.2. Following the fourth lecture, we invited students to voluntarily participate in the experiment by signing a consent form and completing a characterization form. We then used the characterization data to balance the groups, ensuring that the number of participants with expertise in each development area was approximately equal and that the overall experience level was pretty distributed between the groups. The participants' full characterization data is available in our supplementary material[2]. A total of 23 students participated in the experiment, 12 in Group A and 11 in Group B.

The experiment consisted of two assessments conducted on different days. In the first assessment, both groups had access to the guidelines and examples presented in the fourth

[2]https://figshare.com/s/7fd92ace78cbb80a024b

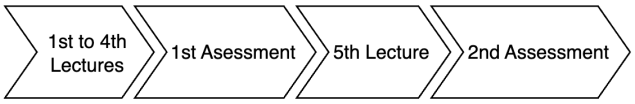

Figure 4. Sequence of steps of the first experiment execution.

lecture. Group A analyzed Object 1, while Group B analyzed Object 2. After completing the first assessment, we delivered the fifth lecture introducing the MFE anti-patterns catalog, which students would use during the second assessment. In the second assessment, Group A analyzed Object 2, and Group B analyzed Object 1, with both groups granted access to the web application containing the anti-patterns catalog. After completing the second assessment, students also completed the feedback form.

5.4 Analysis and Interpretation

Before conducting the data analysis, we excluded responses from students who missed more than one lecture to include only those who participated in the majority of the training sessions. We conducted the quantitative analysis using two groups of paired samples: (1) student assessment scores when consulting practitioner-provided guidelines versus the anti-patterns catalog, and (2) perceived learning scores before and after engaging with the catalog. We used the assessment score samples to test the null hypothesis H_0 and the perceived learning samples to test the null hypothesis H_0'. The raw data is available in our supplementary material.

We began by examining boxplots to visually compare the samples within each group. Next, we applied the Shapiro–Wilk test (Wohlin et al., 2012) to determine whether the samples followed a normal distribution. Based on the results, we selected appropriate paired statistical tests: the paired t-test (Wohlin et al., 2012) for normally distributed samples, and the Wilcoxon Signed Rank Test (Wohlin et al., 2012) for non-normal samples.

Our MFE anti-patterns catalog is available as a web application designed to support MFE development. However, we had not yet analyzed how users interact with it. To investigate how students (representing novice MFE developers) perceived the catalog's usefulness during architectural decision-making in the assessments, we applied the original Technology Acceptance Model (TAM) (Venkatesh and Bala, 2008). TAM assesses users' perceptions of a technology's usefulness and ease of use, which are the two primary factors influencing technology acceptance behavior (Laitenberger and Dreyer, 1998). We measured the constructs of perceived usefulness, perceived ease of use, and behavioral intention using a 5-point Likert scale (Likert, 1932), ranging from "Strongly disagree" to "Strongly agree." In addition to the TAM constructs, we defined four sentences for measuring the catalog's utility for learning MFE. Table 2 presents our adapted version of the TAM questionnaire and the four items related to the catalog's perceived utility for learning.

We collected qualitative feedback on how students used the catalog, its perceived benefits and challenges, how it influenced their learning, and their overall impressions of the catalog. We analyzed the data using Straussian Grounded Theory (GT) procedures (Corbin and Strauss, 2014). We first applied open coding, which involved creating codes representing relevant concepts to understand how students used the catalog. Then, when applying axial coding, we identified connections among the codes and grouped them into broader categories to identify the primary uses of the catalog. We did not apply the GT's selective coding, as our objective was not to develop a theory but to explore how the catalog can be used during MFE development.

Table 2. Adapted TAM constructs and the sentences related to the catalog's utility for learning.

Perceived Usefulness (PU)	
PU01	Using the anti-patterns catalog for MFE improves my performance when developing MFE-oriented architectures.
PU02	Using the anti-patterns catalog for MFE improves my productivity when developing MFE-oriented architectures.
PU03	Using the anti-patterns catalog for MFE improves my effectiveness in communicating with other developers when developing MFE-oriented architectures.
PU04	I consider the anti-patterns catalog for MFE useful for developing MFE-oriented architectures.
Perceived Ease Of Use (PEOU)	
PEOU01	My interaction with the anti-patterns catalog for MFE was clear and understandable.
PEOU02	Interacting with the anti-patterns catalog for MFE did not require much mental effort from me.
PEOU03	I consider the anti-patterns catalog for MFE easy to use.
PEOU04	I find it easy to use the anti-patterns catalog for MFE during the development of MFE architectures.
Behavioral Intention (BI)	
BI01	Assuming I have enough time to design and develop an MFE-oriented architecture, I would use the anti-patterns catalog for MFE.
BI02	Considering that I can choose other tools to assist in the development of MFE-oriented architectures, I intend to use the anti-patterns catalog for MFE.
BI03	I intend to use the anti-patterns catalog for MFE the next time I work on an MFE-oriented architecture.
Useful For Learning (UFL)	
UFL01	I consider the anti-patterns catalog for MFE useful for learning about MFE.
UFL02	I find it easy to use the anti-patterns catalog for MFE to learn about MFE-oriented architectures.
UFL03	The anti-patterns catalog for MFE facilitated learning about best practices in software architecture.
UFL04	The anti-patterns catalog for MFE contributed to my understanding of common challenges in MFE architectures.

6 First Experiment Results

This section presents the results for each RQ-A outlined in Section 5.1. For RQ-A1 and RQ-A2 , we analyze the quantitative data collected during the experiment using boxplots and statistical tests to compare the samples. For RQ-A3 , we present the results of the TAM evaluation and the learning-related statements, and summarize insights from the qualitative analysis of students' feedback. We analyzed the results of students with prior professional experience and found no noticeable differences compared to the other participants.

6.1 Supporting Materials Comparison

To address RQ-A1 , we evaluated the students' scores from the first and second assessments. Figure 5 presents the boxplots for the two samples, which show similar distributions and are close to each other. Performing the Shapiro-Wilk Test on the first sample yielded a p-value of 0.764. The second sample reported its mean, median, and mode as 3.917, 4.000, and 4.000, respectively. Performing the Shapiro-Wilk Test on the second sample yielded a p-value of 0.848. With a significance level of $\alpha = 0.05$, both samples are considered to follow a normal distribution.

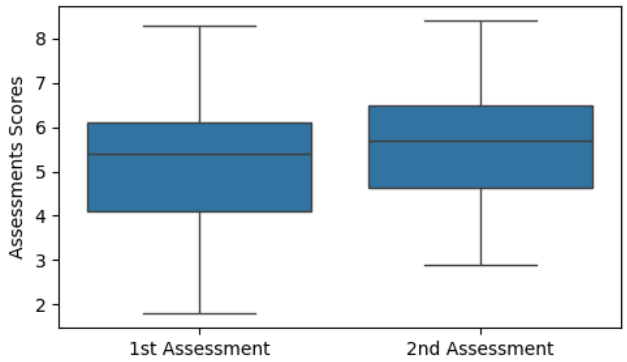

Figure 5. Boxplots presenting the data distribution on the assessment samples.

We selected the paired t-test for the sample comparison, a parametric test for dependent and normally distributed samples (Wohlin et al., 2012). With the significance level set at $\alpha = 0.05$, the test yielded a p-value of 0.298, indicating that we cannot reject the null hypothesis H_0. Therefore, we conclude that both supporting materials are equally effective in helping students learn about MFE.

RQ-A1 Summary

There was no significant difference in students' assessment scores when using the MFE anti-patterns catalog versus the practitioner-provided guidelines. This result indicates that both supporting materials are equally effective in helping students learn about micro frontends.

6.2 Perceived Learning Difference

To address RQ-A2 , we evaluated the students' self-assessed perceived learning before and after engaging with the catalog. Figure 6 presents the boxplots for the two samples, indicating that the second sample appears to have higher values than the first one. Performing the Shapiro–Wilk Test on both samples yielded a p-value of 0.216 for the first sample and < 0.001 for the second sample. Considering a significance level of $\alpha = 0.05$, the second sample is not normally distributed.

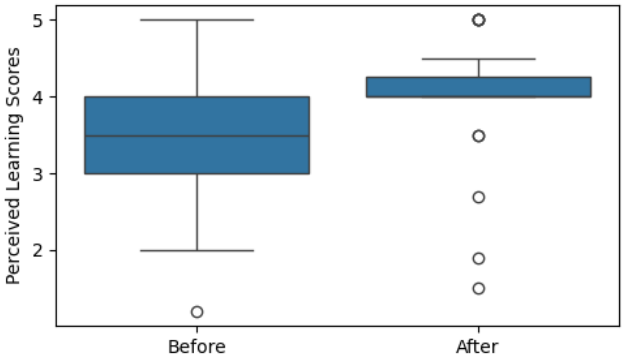

Figure 6. Boxplots presenting the data distribution on the perceived learning before and after engaging with the MFE anti-patterns catalog.

As one of the samples is not normally distributed, we selected the Wilcoxon Signed Rank Test to compare the samples, as it is an appropriate non-parametric test for comparing dependent samples (Wohlin et al., 2012). Using a significance level of $\alpha = 0.05$, the test yielded a p-value of 0.001, allowing us to reject the null hypothesis H_0'. Thus, we can assert a statistically significant difference in students' self-perceived learning before and after using the catalog for solving MFE architectural problems. Since the distribution of perceived learning after engaging with the catalog shows higher values than before, we conclude that using the catalog enhances students' self-perception of learning about MFE.

RQ-A2 Summary

After using the MFE anti-patterns catalog, students reported a statistically significant increase in their perceived learning, suggesting that exposure to real-world problems and examples helps students feel that they have learned more effectively.

6.3 How Students Used the Catalog

To address RQ-A3 , we asked students to answer an online form to respond if they agree with the TAM and the learning-related statements, and we collected qualitative feedback on how students used the catalog, its benefits and challenges, how it influenced their learning, and their overall impressions of the catalog.

6.3.1 TAM and Learning Statements Analysis

Figure 7 summarizes the results on the Perceived Usefulness construct. Upon analyzing PU01, PU02, and PU04, the students generally agreed that the catalog is a useful tool for developing MFE architectures, highlighting its potential to improve their performance and productivity during development. Although they did not create entirely new architectures during the assignment, they were tasked with proposing new

solutions for existing ones. Regarding PU03, some feedback reflected a neutral stance, which may be attributed to the fact that the students did not interact with other developers while proposing their architectural solutions.

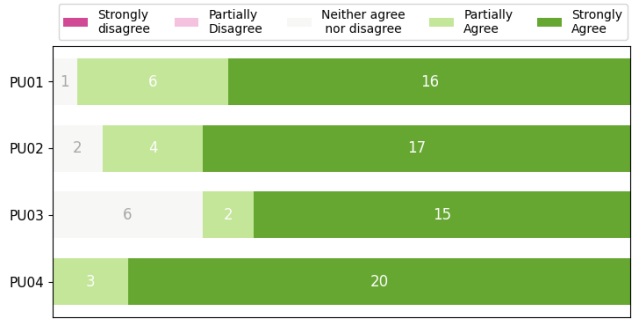

Figure 7. Responses for each sentence in the Perceived Usefulness construct.

Figure 8 presents the Perceived Ease Of Use construct results. Upon analyzing PEOU02 and PEOU03, all the students agreed that the tool was easy to use and did not require much mental effort. Regarding PEOU01, only one response indicated a neutral stance, suggesting that interacting with the catalog was clear and comprehensible. On PEOU04, its feedback indicates that using the catalog to develop MFE architectures may be easy, as it presents neutral responses or partial agreements. These results support the notion that the catalog offers a low-friction experience, which may facilitate its adoption in educational and professional contexts.

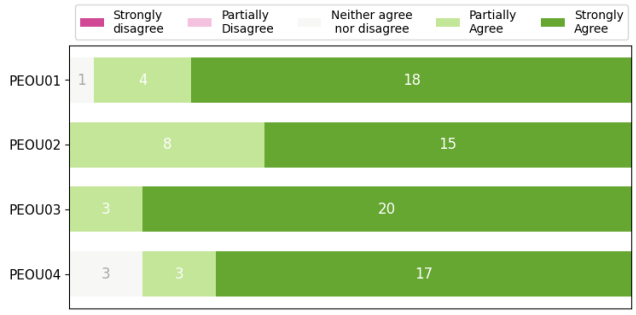

Figure 8. Responses for each sentence in the Perceived Ease Of Use construct.

Figure 9 summarizes the results for the Behavioral Intention construct. Upon analyzing BI01, students generally expressed a positive intention to use the catalog, especially when given sufficient time for design and development tasks. Most participants also agreed they would use the catalog to support MFE development (BI02) and future MFE projects (BI03). These findings suggest that the catalog holds potential as a valuable tool for both in-training software engineers and practitioners.

Figure 10 summarizes the results related to the catalog's utility for learning provided by the students through the survey. Regarding the statements in UFL01, UFL02, UFL03, and UFL04, there were no disagreements or neutral responses, as most were "Strongly agree" and "Partially agree." It suggests that students recognized the catalog's contribution to the learning process concerning software system architecture and MFE architectures. We further discuss the enhancement in learning during the qualitative analysis results.

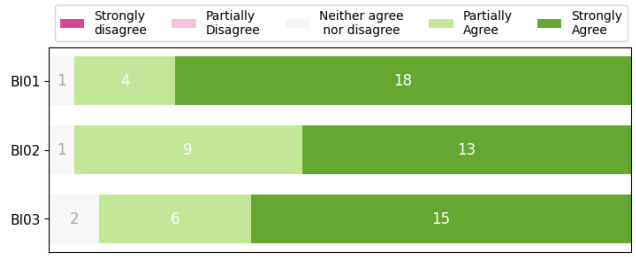

Figure 9. Responses for each sentence in the Behavioral Intention construct.

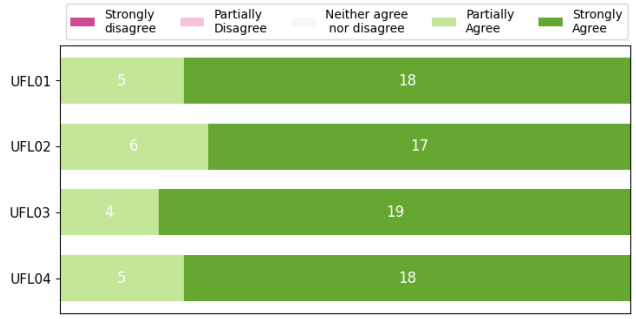

Figure 10. Responses for each sentence related to the catalog's utility for learning.

6.3.2 Qualitative Feedback Analysis

We analyzed students' qualitative feedback using ATLAS.ti[3], applying open and axial coding (Corbin and Strauss, 2014) to understand how students used the catalog during the second assessment. Due to space limitations, the whole coding process and codebook are available in our supplementary material. Based on the relationships among the open codes (highlighted with underlines), we identified three categories that summarize the main ways in which students interacted with the catalog:

Identify problems and solutions – As anticipated, students used the catalog to Identify problems and solutions, as expressed by P1: "*identify the problems and the solutions to the respective questions.*" However, their approaches to identifying problems varied according to the features they relied on within the catalog. Some students emphasized the role of examples in Identify problems by examples, as noted by P3: "*the examples helped me identify the problem,*" and to Understand problems by examples, as stated by P1: "*seeing examples and descriptions helped me understand the problem in some questions.*" Others relied on the visual elements, using the catalog to Understand problems by images, as illustrated by P4: "*the use of images in some explanations helped more than just text.*" Additionally, some students also used the solution description, as seen in P5's comment about to Identify problems by solutions: "*the description of the problems and solutions in the catalog helped me a lot to draw a parallel with the problems proposed.*" Moreover, the catalog also serves to Guide architectural decisions, as stated by P7: "*allowed me to address these problems and made it easier to make decisions.*" These findings suggest that offering

[3]https://atlasti.com/

multiple pathways–textual descriptions, examples, visuals, and solution-based reasoning–can support diverse cognitive strategies among students. Therefore, the catalog enhances accessibility and understanding by accommodating different ways of thinking and learning.

Support efficient and structured search for MFE problems – The catalog's web application enabled a structured and efficient browsing experience, helping students access and explore the anti-patterns more effectively. For example, P7 highlighted the ease of Consulting by categories: *"it is easy to visualize, especially with the use of categories."* Even when facing difficulties in understanding the anti-patterns, P4 used the catalog to Consult several anti-patterns at once and made a suggestion to improve problem identification: *"I had difficulties understanding the description of each anti-pattern; I had to open all of them to see the description. There could be a summary in each card of the catalogs on the main screens."* Similarly, P20 suggested Linking anti-patterns to improve navigability: *"when opening an anti-pattern, the page could show others from the same topic or similar ones."*

Reinforce and deepen MFE knowledge – Students revisited and deepened their understanding of MFE concepts through real-world examples, using the catalog to consolidate and expand their prior knowledge. The catalog can be used to Learn based on practical problems, as P17 remarked: "*Seeing the anti-patterns in practice increased my knowledge about the subject.*" P13 highlighted that the catalog was used to Understand common challenges: "*I could understand real situations and challenges faced during architectural decisions.*" In addition, the catalog was useful to Review MFE concepts, as noted by P21: "*[the catalog] helped me remember some concepts I did not recall.*" P4 reinforced this point and suggested improvements to enhance its role as a learning tool by adding the MFE concepts presented in class: "*[the catalog] helped by centralizing content about MFEs. However, the catalog could include summaries and slides presented in the classes.*" These insights demonstrate the catalog's potential as a reference material and an educational tool that reinforces learning through contextualized, practice-based content.

RQ-A3 Summary

Students agreed that the MFE anti-patterns catalog is useful, easy to use, and helpful for learning about MFE. They also expressed an intention to use it in future development tasks. Qualitative analysis revealed three primary ways students used the catalog: (1) to identify problems and solutions, (2) to support efficient and structured searches for MFE issues, and (3) to reinforce and deepen their understanding of MFE concepts. These findings highlight the catalog's effectiveness as a problem-solving aid and an educational resource that accommodates different learning strategies.

7 Catalog Improvement

After the first experiment, we improved the catalog based on students' feedback by adopting a new template to reduce the amount of text and adding more images. We also expanded the catalog by adding 15 new anti-patterns identified in grey literature, including *Avoiding Observability* (Rappl, 2024), *Access to Different Domains* (Wessels, 2020), *Bidirectional Data Flow* (Mezzalira, 2020), *Chatty Micro Frontends* (Rappl, 2024), *Dependency Hell* (Mezzalira, 2020; Shinde, 2022; Matteo Figus and Mezzalira, 2024; Raimundo, 2023), *Dependent Deployment* (Rappl, 2024; Casas, 2020; Matteo Figus and Mezzalira, 2024), *Dismissing Human Factors* (Rappl, 2024), *Distributed Data Inconsistency* Rappl (2024), *Framework Frenzy* (Rappl, 2024; Mezzalira, 2020; Shinde, 2022; Mezzalira, 2021c; Raimundo, 2023), *Global State Communication* (Mezzalira, 2020; Shinde, 2022; Mezzalira, 2021b,c; Raimundo, 2023), *Hammering APIs* (Mezzalira, 2020), *One Micro Frontend for All* (Gkamperlo, 2020), *Partial UI Migration* (Mezzalira, 2024), *Spaghetti Architecture* (Rappl, 2024), and *Unmediated Legacy Integration* (Mezzalira, 2020; Raimundo, 2023). The complete catalog is available in the supplementary material and through the catalog's web application.[4]

As the catalog grew in both size and complexity, we adopted an extended template to document the anti-patterns. Drawing on the Full AntiPattern Template (Brown et al., 1998), the general template used in the C2 Wiki repository (Cunningham, 2013), and the template proposed by Brada and Picha (2019), we defined a new anti-pattern template that includes additional fields: **Solution Pitfalls**, **Related Anti-Patterns**, **References**, and **Also Known As**. We also restructured the problem and solution descriptions. We divided the original **Problem** field into two fields, **Problem** and **Symptoms and Consequences**, to simplify the problem definition and highlight its consequences. Similarly, we split the original **Solution** field into **Solution** and **Resulting Context** to describe the solution and its potential benefits or risks separately. Figure 11 illustrates the *Global State Communication* anti-pattern using the new template fields.

8 Second Experiment Design

After evolving the catalog based on the results and feedback from the first experiment, we formulated a new set of research questions (Section 8.1) to compare the two catalog versions through a second controlled experiment. The following subsections describe each aspect of the second experiment in detail.

8.1 Goal and Research Questions

This experiment aims to compare the two versions of the MFE anti-patterns catalog. We also investigate which factors influence participants' preferences between the two versions and explore how developers and students can use the catalog as a learning resource for MFE. Based on these goals, we define a new set (B) of research questions:

[4] https://mfe-anti-patterns.online/micro-frontends-anti-patterns/#/catalog

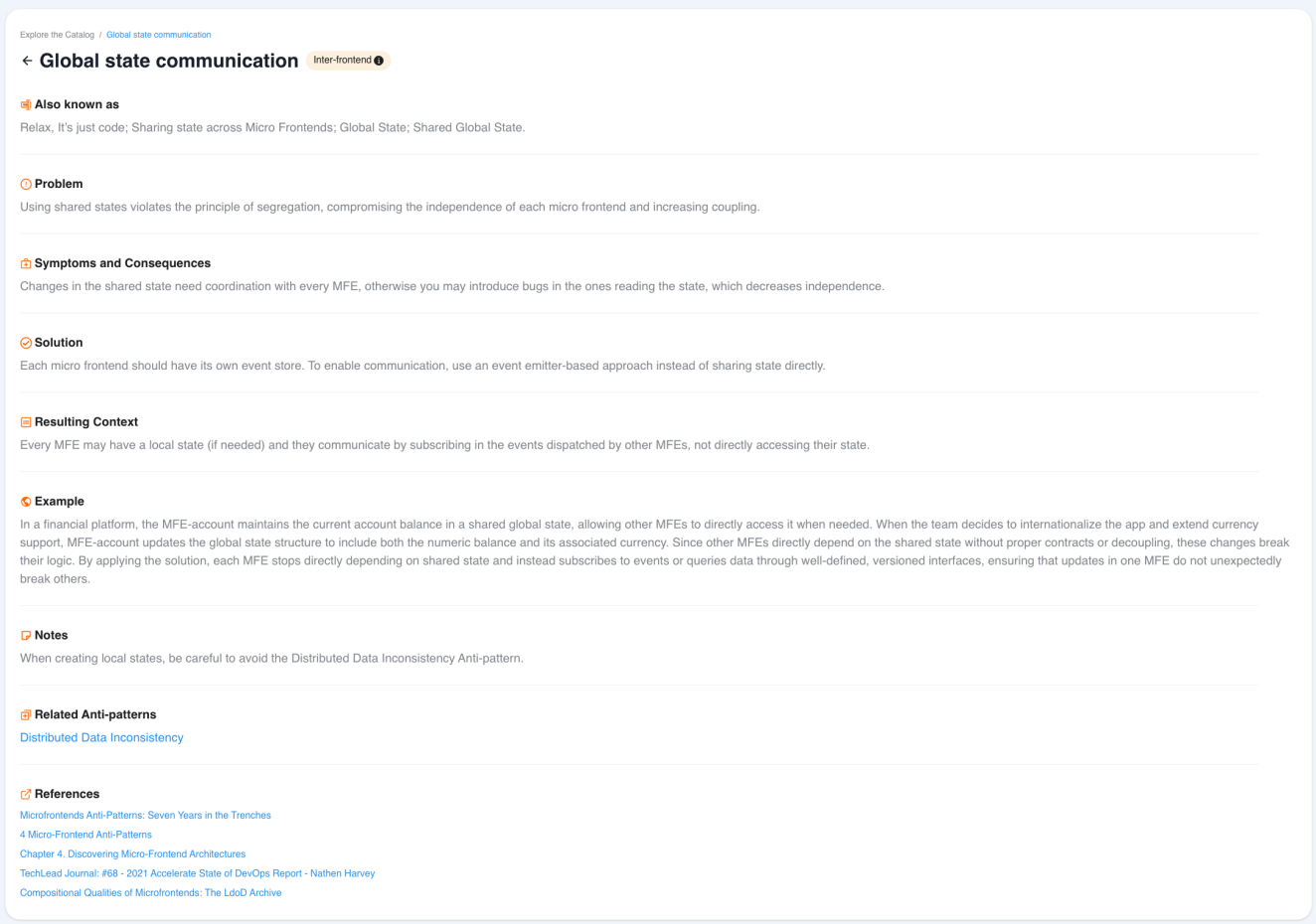

Figure 11. Global State Communication anti-pattern description from the web application of catalog's second version.

RQ-B1

Which version of the Micro Frontend Anti-patterns Catalog more effectively supports novice developers when solving MFE-related problems?

- RQ-B1.1: Which version leads to higher student assessment scores?
- RQ-B1.2: Which version do students prefer?

To answer RQ-B1 and evaluate whether the changes we introduced truly improved the catalog, we decomposed the research question into two sub-questions. To address RQ-B1.1, we compared the mean scores of the MFE assessments completed by students when using the first and second versions of the catalog. Finally, to address RQ-B1.2, we asked students to rate each catalog version on a scale from 0 to 10 and compared the mean ratings.

In addition to identifying which catalog version students prefer, we defined RQ-B2 to understand which characteristics of supporting tools influence developers' preferences. We also formulated two additional research questions to further explore how developers and students can use the catalog as a learning tool (RQ-B3) and to search for architectural problems within it (RQ-B4). To address RQ-B2, RQ-B3, and RQ-B4, we analyzed the qualitative feedback provided by students after using both catalog versions through GT procedures (Corbin and Strauss, 2014).

RQ-B2

What factors influence novice developers' preferences for each version of the Micro Frontend Anti-patterns Catalog?

RQ-B3

How can the Micro Frontend Anti-patterns Catalog support learning about MFEs?

RQ-B4

How do novice developers search for architectural problems using the Micro Frontend Anti-patterns Catalog?

8.2 Planning

We planned the second experiment according to the guidelines proposed by Wohlin et al. (2012). Since we conducted this experiment with students enrolled in the same course as in the first experiment, we adopted the same context and participant selection described in Section 5.2.1 and Section 5.2.4, respectively.

8.2.1 Variable Selection

The independent variable is the version of the MFE anti-patterns catalog used during the assessments, with two treatments: (1) the first version, the same used in the first experiment (see Section 3.1), and (2) the second version, which incorporates updates based on participants' feedback from the first experiment (see Section 7). The dependent variables are the students' assessment scores and the ratings assigned to each catalog version. All scores are real values ranging from 0 to 10, allowing statistical comparisons between samples (Wohlin et al., 2012).

8.2.2 Hypothesis Formulation

We formulated three null hypotheses, each corresponding to a sub-question on RQ-B1: H_0'': There is no significant difference in students' mean assessment scores when supported by each version of the MFE anti-patterns catalog (RQ-B1.1); and H_0''': There is no significant difference in students' mean version scores when supported by each version of the MFE anti-patterns catalog (RQ-B1.2). Each hypothesis has an alternative hypothesis H_A, which is its negation and is accepted if we can not reject the null hypothesis. As RQ-B2, RQ-B3, and RQ-B4 are addressed through qualitative methods, we did not define hypothesis for them.

8.2.3 Study Design

Students completed the two assessments described in Section 4.3 using the same Objects 1 and 2 from the first experiment. However, in this experiment, we applied a crossover design (Vegas et al., 2015) to the treatments rather than to the objects. We did not inform the students that they would use different catalog versions. Instead, we only stated that they would analyze two architectures in two separate assessments.

To enable the crossover design, we divided the students into two groups: Group A and Group B. Since none of the students had prior experience with MFEs, we balanced the groups by type of software development experience (backend, frontend, or full-stack) and, when applicable, by its duration. During the first assessment, both groups analyzed Object 1, but each group used a different catalog version: Group A used version 1, and Group B used version 2. In the second assessment, both groups analyzed Object 2 with the catalog versions reversed: Group A used version 2, and Group B used version 1. Figure 12 summarizes the study design.

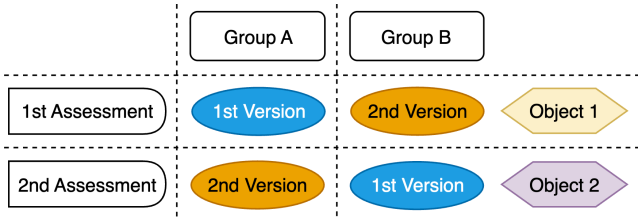

Figure 12. Crossover design adopted in the second experiment.

8.2.4 Instrumentation

We reused the same instruments from the first experiment described in Section 5.2.6. To ensure compatibility between the two catalog versions, we incorporated the new anti-patterns into the initial version while preserving its original structure. We also added one question to the assessments related to the *Global State Communication* anti-pattern to cover at least one of the newly added anti-patterns. Additionally, we introduced two new instruments:

1. **Feedback Forms**: We created two feedback forms, each applied after one of the assessments. Both forms include open-ended questions to evaluate the catalog version used in the corresponding assessment, asking students to provide improvement suggestions and list advantages and disadvantages. Each form also includes a rating question in which students rate the catalog version on a scale of 0 to 10. The second form also includes open-ended questions that explicitly ask students to compare the two versions and indicate which they would recommend and why.
2. **Catalog Web Applications**: To avoid interfering with the publicly available catalog, we deployed two separate web applications, each hosting one version of the catalog: Version 1[5] and Version 2[6]. We also implemented a logging script to capture user interactions with the catalog's main features, such as opening pages, clicking catalog cards or category filters, and performing search queries.

8.3 Execution

Since we reused the same instruments from the first experiment and two experienced MFE professionals had already reviewed them, we did not conduct a new pilot study. Before executing the experiment, we delivered four instructional sessions covering the content described in Subsection 4.2. We merged the last two sessions so we could introduce the anti-patterns and demonstrate how to evaluate real architectures using some of them.

As the catalog now contains 27 anti-patterns, we could not present all of them during the sessions. Instead, we sent students a document containing the name, problem, and solution of each anti-pattern before the first assessment. It allowed students to study the anti-patterns without interacting with any version of the catalog beforehand.

Following the fourth session, we invited students to voluntarily participate in the experiment by signing a consent form and completing a characterization form. We then used the characterization data to balance the groups. No student had prior experience with MFEs. However, four students had prior experience as full-stack developers (P2, P11, P12, and P22), and one had experience as a frontend developer (P5). All reported experiences lasted less than one year. We first randomly assigned these five participants to groups, then assigned the remaining participants. This process resulted in 16 participants in Group A and 13 in Group B, for a total of 29 participants. The participants' full characterization data is available in our supplementary material.

To avoid cross-group contamination, we split students into their respective groups across two laboratories, ensuring

[5] https://github.com/nabsonp/mfe-ap-catalog-version-1
[6] https://github.com/nabsonp/mfe-ap-catalog-version-2

each group had access only to its assigned catalog version during the assessment. The experiment consisted of two assessments conducted in consecutive weeks to mitigate learning effects (Wohlin et al., 2012). Each assessment lasted up to two hours. After completing each assessment, students also filled out a feedback form.

8.4 Analysis and Interpretations

Before conducting the data analysis, we excluded responses from students who missed the last session, in which we presented additional anti-patterns and discussed real MFE architectures. Because this session directly supported the assessment tasks, we did not include students who missed it in the analysis. Unlike the first experiment, we did not consider the answers to the first assignment question, in which students evaluated the proposed architecture. During the analysis, we observed that several students interpreted this question as requiring a brief evaluation of the architecture rather than identifying as many issues as possible. Since this interpretation did not align with the task's intended objective, we excluded this question from the analysis to avoid introducing noise into the results.

We conducted the quantitative analysis using paired samples to test the null hypotheses: (H_0'') differences in student assessment scores when using each catalog version, and (H_0''') differences in the scores students assigned to each version. As in the first experiment, we began the analysis by visualizing the samples with boxplots and then tested normality using the Shapiro–Wilk test (Wohlin et al., 2012). Based on the normality results, we compared the samples using the appropriate statistical test: the paired t-test for normally distributed samples and the Wilcoxon signed-rank test for non-normal samples (Wohlin et al., 2012).

We analyzed the qualitative data using the same GT procedures (Corbin and Strauss, 2014) adopted in the first experiment. We reused the codes identified in the first study to detect recurring or new patterns in the data, thereby extending and deepening the previously conducted qualitative analysis. We also created memos to explain and summarize the relationships between codes, preserving the reasoning behind the analysis.

To better interpret each student's feedback, we analyzed their navigation behavior across the catalog versions using the interaction logs collected during the assessments. We generated a chart in which each horizontal bar represents the time a screen remained focused in the browser. We excluded time spent on unfocused activities, as students were likely completing the assessment rather than interacting with the catalog. We also plotted markers indicating when students used the search bar or clicked on a category filter. We produced these visualizations for each catalog version to enable comparisons of participants' interactions across versions. Figure 13 presents an example of the navigation chart for one catalog version.

9 Second Experiment Results

This section presents the results for each RQ-B outlined in Section 8.1.

9.1 Versions Comparison

To answer RQ-B1, we first address RQ-B1.1 and RQ-B1.2.

Assessment Scores – To address RQ-B1.1, we analyzed the scores students obtained in the two assessments when using the first and second catalog versions. Figure 14 presents the boxplots for the two samples. For the first version, students obtained a mean score of 7.16 and a median of 7.5. For the second version, the mean score was 7.49, and the median was 8.0. Although the second version shows a slightly higher median score, the substantial overlap between the distributions suggests similar performance when using both catalog versions. The Shapiro–Wilk test yielded a p-value of 0.427 for the first sample and 0.047 for the second sample. Considering a significance level of $\alpha = 0.05$, only the first sample follows a normal distribution, while the second sample deviates from normality.

Since one of the samples deviates from a normal distribution, we used the Wilcoxon signed-rank test to compare them (Wohlin et al., 2012). With a significance level of $\alpha = 0.05$, the test yielded a p-value of 0.692, indicating that we cannot reject the null hypothesis H_0''. Therefore, the results do not indicate a statistically significant difference between the scores, suggesting that both catalog versions supported students similarly when solving MFE problems.

Version Ratings – To address RQ-B1.2, we analyzed the ratings students assigned to each catalog version after using it to complete an assessment. Figure 15 presents the boxplots for the two samples, which show very similar distributions, with ratings concentrated in the upper range of the scale. For the first version, students reported a mean rating of 8.97 and a median of 9.0. The Shapiro–Wilk test yielded a p-value of 0.002. For the second version, the mean rating was 9.29 and the median was 9.5, with the Shapiro–Wilk test yielding a p-value of 0.0001. Considering a significance level of $\alpha = 0.05$, both samples deviate from a normal distribution.

Since both samples deviate from a normal distribution, we used the Wilcoxon signed-rank test to compare them (Wohlin et al., 2012). With a significance level of $\alpha = 0.05$, the test yielded a p-value of 0.176, indicating that we cannot reject the null hypothesis H_0'''. Therefore, the results do not indicate a statistically significant preference for either catalog version, suggesting that students perceived both versions positively and rated them similarly.

RQ-B1 Summary

The results show no statistically significant differences in either the assessment scores or the ratings assigned by students to the two catalog versions, suggesting that both versions similarly support students when solving MFE-related problems.

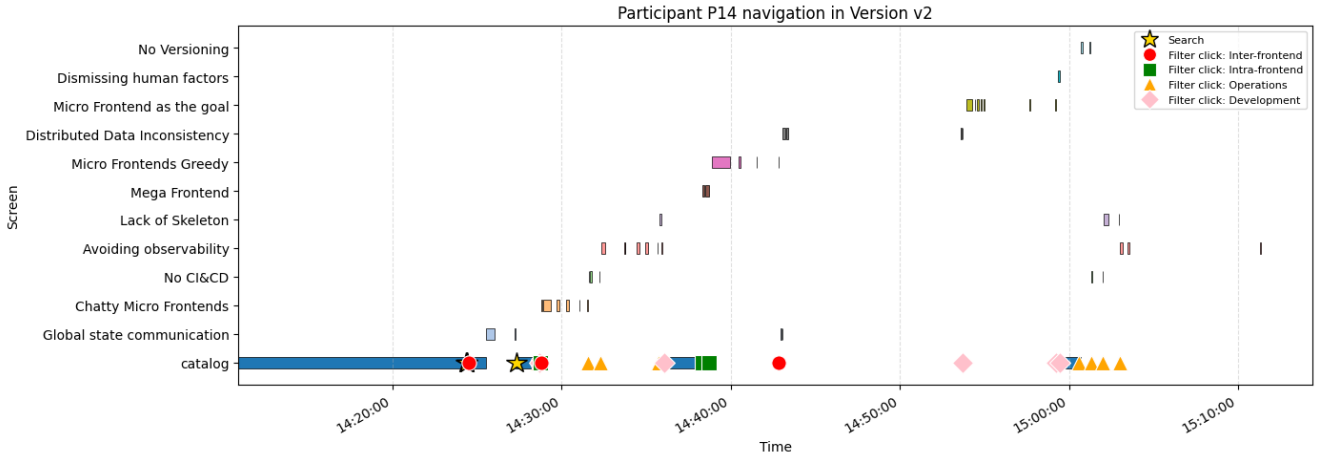

Figure 13. Dashboard summarizing the main actions performed by P14 while interacting with one version of the catalog. Initially, P14 focused on reading the anti-pattern descriptions displayed as cards on the main screen (catalog). Then, after performing a search and clicking the category filters, P14 mainly focused on reading the full anti-pattern descriptions on the details screen, without spending much time on the main screen again. The blank spaces between rectangles indicate periods in which the participant was answering the activity, often returning to the anti-pattern details screen to review the information.

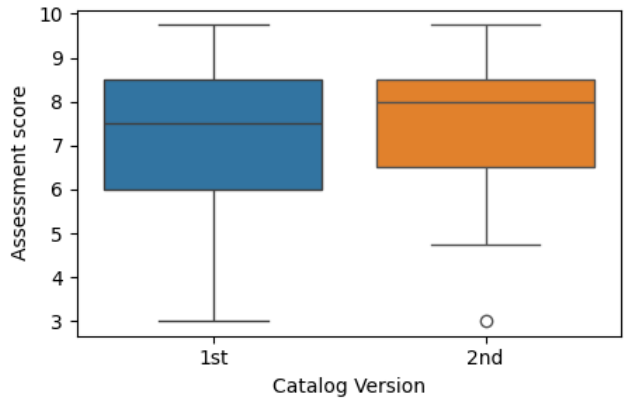

Figure 14. Distribution of assessment scores obtained when students used the first and second catalog versions.

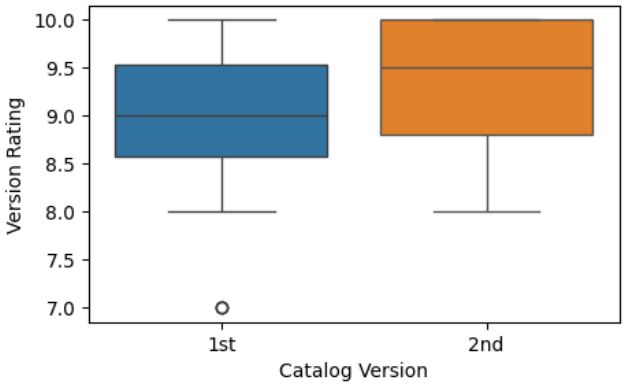

Figure 15. Boxplots presenting the data distribution on each version ratings.

9.2 Factors Influencing Version Preference

After observing no statistically significant differences between version 1 (see Figure 1) and version 2 (see Figure 11) of the catalog, we analyzed participants' feedback to answer RQ-B2 and understand the factors that influenced their preferences. We identified three main factors influencing these preferences, as well as some students who reported no clear preference for either version.

Preference for visual elements - Some participants tended to favor the version that included more images. For instance, explicitly preferred Version 2 (V2) because it contained a code-related image that was not present in Version 1 (V1): "I prefer the version from the previous session (V2). It had a code image that I could not find today (V1)."

Visual elements helped participants understand the anti-patterns more easily, making the explanations clearer and the reading more dynamic. P4 highlighted this aspect when recommending the second version: "I would recommend the second one (V2) because it is more direct and more explanatory due to some images that it includes, which were not present in the previous version (V1)". Similarly, P26 noted that the images improved readability: "I believe the images made the reading better and more dynamic compared to the other version."

P6 also emphasized the importance of visual support. While this participant struggled to understand some examples presented only in text, they reported understanding all examples that included images: "Some examples and solutions were not very clear to me, but I managed to understand all the ones that had a figure as an example". The same participant also suggested adding more images to facilitate faster comprehension: "Using more images would help understand some anti-patterns faster, without the need to change the text."

Preference for less segmented text – Although the second catalog version adopts a more structured template to describe the anti-patterns, some participants preferred the less sectioned text structure used in the first version because they perceived it as more direct, presenting the problem and solution immediately. P25 expressed this perception when referring to V1: "I think the advantage is that the information is more centralized, and the problem and solution appear right away." From this perspective, the stronger text segmentation in V2 became a disadvantage for some students. P12 commented that V1 seemed more explanatory: "I have the impression that the other version (V1) presented more explanatory texts. If that is true, I think the brevity of the texts in this version (V2) is a disadvantage."

Other participants also preferred the more direct style of V1. For example, P3 stated: "I would recommend the current

one because it is more direct and more descriptive in the examples." Similarly, P7 highlighted that the first version presented the anti-patterns more directly: "I would recommend the first version more. I noticed the patterns were more direct." The same participant also felt that V2 lacked direct descriptions and required more time to understand: "I missed some more direct descriptions," and "I feel it took longer to understand."

P17 also expressed a similar preference for V1, stating: "I preferred the second one (V1); I found it more robust, and some explanations seemed more detailed." The interaction logs support this perception. The participant showed similar navigation behavior in both versions, opening a comparable number of anti-patterns and making minimal use of filters (one click in each version) without using the search feature. However, in V2 the participant spent more time reading the same anti-pattern, particularly *Cyclic Dependency*. In contrast, they navigated through the same anti-pattern more quickly in V1, suggesting that the participant required more effort to interpret the information in the more segmented structure of V2.

Preference for more segmented text – On the other hand, some students perceived the segmentation of the text into smaller sections as an advantage of V2 because it improved organization and readability. Shorter sections made the content easier to read and navigate, as P4 highlighted by stating: "I felt the reading was easier because of the division into more sections and shorter texts." Similarly, when asked about the advantages of V2, P22 commented: "Yes, because I found it better structured, especially the examples, solutions, and causes." From this perspective, the longer descriptions in V1 sometimes took longer to read and understand. For instance, P6 reported: "Maybe the problem I had was the time it took me to understand what was written. Maybe the text of some anti-patterns is too long." In the same direction, P20 described V2 as clearer and more concise: "This one (V2) seems better, explained in a more summarized way."

Other participants also associated the V2 template with clearer information organization. P24 noted: "I found it more organized than the previous one, and the examples seemed more direct. It also related the anti-patterns to each other." For this reason, the same participant recommended V2: "The current one (V2), because it is more organized and more direct." This perception contrasts with the previous factor, as participants may interpret what is "more direct" differently; some prefer centralized explanations, while others find segmented structures easier to follow.

Additionally, some students highlighted the usefulness of version 2 sections that connect anti-patterns to related ones or provide references. These elements helped them identify secondary issues and better understand the relationships between architectural problems. For example, P14 preferred V2 "because of the related anti-patterns and the examples." Similarly, P10 explained that "the related anti-patterns help the understanding because we start connecting one thing to another." P14 also noted that "the advantage of V2 is showing related MFEs that may occur together, which helps identify secondary or less obvious problems." In contrast, P25 mentioned missing this feature when returning to V1 after using V2: "the other topics were also interesting, especially the part that related how one anti-pattern could lead to another." Finally, some participants highlighted the usefulness of the *Symptoms and Consequences* section in V2. P26 explicitly mentioned that "the symptoms were also very useful."

No clear preference – Some participants did not express a clear preference between the two catalog versions. These students reported not noticing meaningful differences among themselves. For example, P15 stated: "To be honest, I did not notice any differences." Similarly, P5 mentioned that both versions could be recommended: "Both, I did not notice any difference between them." P13 also reported the same perception: "I did not notice differences between the versions."

P28 suggested that the preferred version may depend on the user's goal or context. For instance, P28 preferred V2 because it contained more information, explaining: "I think the first one. There is more information, which may be interesting for someone who really wants to understand and identify the problem." However, the same participant also pointed out that the most suitable version depends on the reading purpose. According to P28, V1 may be more advantageous for quick consultation because "it is more summarized, which is better for quick reading." Based on this perspective, P28 suggested combining characteristics of both versions. The participant proposed adding a feature that allows readers to expand the content when needed: "To keep a middle ground between the two versions, a button to show more information could be useful." The same idea was reinforced by suggesting interface controls such as "See more" or "See examples" buttons that would reveal the additional sections available in the other version. These comments suggest that, rather than choosing a single superior version, some users may benefit from a hybrid design that supports both quick consultation and deeper exploration of the anti-pattern descriptions.

RQ-B2 Summary

Students tended to prefer the version that best matched their reading style. Some favored the first version for its more direct, concise descriptions, while others preferred the second for its more structured organization and visual elements. However, some participants reported no clear preference and suggested that access to both versions could be beneficial depending on the task.

9.3 How the Catalog Can Support MFE Learning

To answer RQ-B3, we analyzed the participants' feedback to understand how students and developers can use the catalog to support learning about MFEs. These ways of using the catalog were already observed in the first experiment, but this study allowed us to explore further and deepen this understanding. We discuss each way in the following.

Learning from real-world examples – As already observed in the first experiment, exposure to realistic architecture examples helps students understand how MFE problems manifest in practice and how they can be resolved. For instance, P25 stated that "the catalog is great for understanding the core of the problem, how it starts, or what its symptoms

are." Similarly, P13 mentioned that the anti-patterns were "very useful to understand the consequences of some problems and the solutions that have already been implemented." These comments indicate that the catalog examples help students understand not only the problem itself, but also its consequences and possible solutions.

Other participants emphasized that practical examples help consolidate theoretical knowledge. P24 explained that "having practical examples of the error and the solution helps more to synthesize and retain the concepts." Likewise, P2 reported that "the problems and solutions they present are very useful to understand system problems and solve them in a more thoughtful and efficient way." In the same direction, P15 noted that the anti-patterns "help in correctly perceiving the problem being presented."

Learning from architectural mistakes – Several students reported that learning from architectural mistakes helped them better understand MFE concepts. For example, P12 explained that "knowing what not to do helps to understand the goals of this technology." Similarly, P3 noted that "by studying and analyzing the possible errors in implementing MFEs, it is possible to have a better notion of what ideal MFEs would look like." In the same direction, P4 explained that learning only the correct approach was not enough: "Even knowing what would be a 'correct' MFE, before I would not be able to differentiate a 'wrong' MFE and would remain in doubt."

Participants also highlighted that anti-patterns clarified architectural concepts such as Domain-Driven Design (DDD). P4 explained that seeing what not to do made the concept clearer: "If it were only a theory saying 'do it this way', I believe I would still have a greater chance of making the mistake of creating MFEs that are not based on subdomains. Now, with the anti-patterns (the 'DO NOT do it this way'), the concept of DDD became much clearer." Students also reported that anti-patterns function as a practical guide for architectural decision-making. P27 stated that they provide "greater ease when modeling a micro frontend architecture with a guide on what not to do and how to know when to split into a new micro frontend." Likewise, P28 emphasized that "it is important to know the possible mistakes in order to start with a good architecture."

Finally, participants noted that learning from these mistakes reinforced the importance of making sound architectural decisions. P25 explained that anti-patterns "demonstrate actions that may be taken impulsively or without understanding how certain decisions can generate catastrophic problems in the application." Although identifying anti-patterns may require additional effort, some students considered this effort worthwhile compared to fixing problems later. As P28 described, "being willing to do this is better than running after the damage caused by falling into several anti-patterns later," and the catalog "helps identify problems before they even exist, giving us the possibility to avoid them."

Deepening knowledge after initial learning – Some students reported that the catalog works better as a second step in the learning process rather than as an introductory resource for MFEs. For example, P9 described the catalog as "a great second step in the MFE learning process, because you first understand what MFEs are and then understand how they can be implemented in wrong ways." The same participant also noted that "I do not think it necessarily helps to explain the concepts of MFEs."

Other students mentioned that some technical terms may make the catalog harder to use without prior knowledge. P20 pointed out that "some terms and examples could be more specific. Someone who does not have much programming experience may have difficulty understanding the content because they may not know the concepts." Similarly, P24 explained that unfamiliar terminology can make it harder to identify the correct anti-pattern: "If the person does not fully master the technical terms, they may become confused about which anti-pattern their problem fits best." These comments suggest that the catalog is more effective when used after an initial introduction to MFE concepts, functioning as a resource to reinforce knowledge and help students reason about incorrect architectural decisions.

Reviewing MFE concepts – This way of using the catalog had already emerged in the qualitative analysis of the first experiment and was reinforced in the second experiment. Participants in the second experiment reported that the catalog helped them recall concepts previously learned in class. For example, P26 explained that "the definitions help to recall the concepts." Similarly, P21 stated that "the anti-patterns were very useful to understand the concepts seen in class." These comments suggest that the catalog can help students revisit key MFE concepts after initial exposure to the topic. However, this perception was not unanimous among participants, as P9 previously argued that the catalog does not necessarily help explain MFE concepts.

Participants also highlighted that having the catalog available after the lectures helps them confirm their understanding and quickly access solutions. As P26 noted, "even having some notion of the mistakes from the lectures, having the catalog always to have the solution at hand is very useful." This indicates that the catalog can serve as a lightweight reference for students to review concepts and confirm architectural decisions.

RQ-B3 Summary

Students reported multiple ways of using the catalog to support learning about MFEs. They use it to learn from realistic examples of architectural problems and their solutions, to understand the consequences of "wrong" architectural decisions through anti-patterns, to deepen their understanding after learning the basic concepts of MFEs, and to review concepts previously learned in class. Together, these uses suggest that the catalog supports different stages of the learning process, from reinforcing concepts to helping students reason about architectural mistakes in practical scenarios.

9.4 How Developers Search for Problems

To answer RQ-B4, we analyzed participants' feedback together with the navigation charts generated from the interaction logs to understand how they interacted with the catalog when searching for architectural problems and potential

solutions. In the first experiment, we identified initial ways developers use the catalog. In this study, we extend this analysis to better understand their interaction strategies, which can inform improvements to the catalog and the design of new features or supporting tools. Our analysis revealed three main strategies developers use to search for problems in the catalog:

Comprehensive search – Some students searched the catalog by browsing several anti-pattern cards on the home page before opening their detailed descriptions. P4 reported using the snippets displayed on the home page to filter possible matches: "the snippet also helped a lot to filter the possible anti-patterns applicable to the problems." Similarly, P5 explained that repeatedly opening detailed pages interrupts the search process: "I have to click a button to see the whole content. It is a bit annoying when you want to search for something and keep being redirected to another page."

These behaviors indicate a comprehensive search strategy, in which users scan multiple anti-patterns until identifying one that resembles the problem being analyzed. This behavior aligns with the code *Consult several anti-patterns at once* identified in the first experiment. In this strategy, the organization of the catalog and the information available directly on the home page play an important role in supporting exploration. For instance, P4 highlighted that "alphabetical ordering helped a lot to find some anti-patterns whose names I did not remember well." In these cases, participants relied heavily on anti-pattern names to identify potential matches. P5 described recognizing a problem and immediately associating it with a known anti-pattern name: "when I see, for example, a problem about the difficulty of creating a new MFE due to the lack of standards, I immediately think about 'Lack of Skeleton'."

Selective search – Some students preferred a more direct strategy to locate relevant anti-patterns, relying on search mechanisms rather than browsing multiple catalog cards. Participants with this behavior used the catalog search field or the browser's search functionality (e.g., CTRL+F) to locate terms related to the problem they were analyzing quickly. P5 highlighted how keyword search simplified the process: "just press CTRL-F and type the keyword and I already get what I want."

This behavior indicates a more selective search strategy, in which users attempt to quickly identify the most relevant anti-pattern instead of exploring multiple catalog entries. In this context, participants suggested that more advanced search mechanisms could further support this objective-oriented interaction. For example, P9 proposed using Artificial Intelligence (AI) based assistance to analyze architectural context and suggest likely anti-patterns: "we provide the context and the problem of our system, and then an AI infers or ranks the anti-patterns we are most likely committing." Similarly, P7 suggested implementing similarity-based search mechanisms: "a similarity search based on user input. For example: Problem X -> return the anti-pattern with the highest score for that problem."

Categorical search – Some participants reported using the catalog categories to narrow the set of candidate anti-patterns before exploring them in more detail. P4 explained that the "separation into four sections helped because of the division of anti-pattern types." Similarly, P6 mentioned focusing mainly on specific categories while searching for problems: "I stayed mostly focused on the inter-frontend ones." P12 also highlighted that the visual category indicators supported reading and navigation, stating that "the colored category identifiers made reading easier."

Participants also noted that some categories were easier to understand than others. P4 pointed out that categories such as "inter and intra frontends were more intuitive." However, the usefulness of this strategy depended on understanding what each category represents. As P26 observed, "if I did not already have a notion of what each category means, such as inter or intra frontend, it would have been harder to locate the problems."

RQ-B4 Summary

Students searched for architectural problems in the catalog using three main strategies: browsing several anti-patterns to identify possible matches, directly searching for specific terms, and filtering candidates through category selection. These strategies reflect distinct information-seeking behaviors, suggesting that architectural knowledge tools should support multiple navigation mechanisms to assist different user types effectively.

10 Threats to Validity

We evaluated the validity of the experiment results based on the four types of threats to validity defined by Wohlin et al. (2012). The threats discussed in this section apply to both controlled experiments, as they share the same educational context and experimental procedure.

Internal Validity: To mitigate bias from students' unfamiliarity with MFE, we conducted sessions about MFE fundamentals before the assessments. We also measured students' perceived learning after the sessions and before engaging with the catalog to isolate its impact on the students' perceived learning. To avoid bias from sample characteristics influencing treatment responses, we balanced groups based on development experience. To prevent one group from replicating the other's treatment, we placed groups in separate labs, restricted object access, and renamed the object in the second assessment. To address learning effects, we used a crossover design. Lastly, we ensured both lectures enabled participants to answer all questions to avoid training bias. In the second experiment, the new version introduced additional anti-patterns, which could influence the results independently of the structural changes made to the catalog. To mitigate this threat and ensure a fair comparison, we included the newly identified anti-patterns in both versions of the catalog while preserving the original structure of the first version.

External Validity: Interaction of setting and treatment is a potential threat, as the experimental objects may not fully reflect real-world architectures. To address this, we designed realistic scenarios based on professional experience and validated them with MFE-experienced practitioners. Another threat is that participants may not represent real developers.

To mitigate this, we used sixth-semester students, provided training, and excluded data from those who missed more than one lecture, approximating novice developers with limited MFE experience.

Construct Validity: A potential threat is that the measure may not accurately reflect its intended effect. To address this, we conducted a pilot study with two practitioners whose feedback aligned with our correction criteria, supporting its reliability. To reduce experimenter expectancy bias, we based questions on real-world problems and validated them in the pilot, avoiding direct use of catalog examples. We also adopted a crossover design and conducted treatments in separate weeks to mitigate carryover effects. Finally, we withheld the origin of the catalog to prevent hypothesis guessing and reduce biased feedback. Comparing two catalog versions may introduce learning bias, as students could become familiar with the task after using the first version. To mitigate this threat, we adopted a crossover design in which students used the versions in different orders. We also observed little difference between the groups, suggesting that the order had minimal impact on the results.

Conclusion Validity: We mitigated low statistical power by using paired statistical tests and adopting a crossover design, which allowed us to collect more data per participant and increased the power of our analyses. Regarding the risks of fishing and error rates, one object might favor one treatment over the other, biasing the results. We designed both assessment objects with equivalent architectures and comparable problems to mitigate this.

11 Discussion

In this section, we discuss the implications of our findings for the design of architectural support tools and implications for researchers.

11.1 Understanding How Users Interact with Architectural Support Tools

Our results highlight the importance of studying how students and developers actually use educational and architectural support tools. It resonates within Razavian et al. (2019) findings, as the authors emphasize the role of tools and methods designed to support developers during architectural decision-making. Without understanding usage patterns and user preferences, it becomes difficult to design tools that effectively support architectural reasoning and that are likely to be adopted in real-world settings.

The navigation charts show that students adopted different strategies to navigate the anti-pattern catalog when searching for architectural problems. Some participants preferred a more exploratory strategy, browsing the catalog through the home page and scanning the available anti-patterns until they found one that resembled the problem they were facing. This behavior resembles a more comprehensive information-processing style (Anderson et al., 2024), in which users prefer to explore available information. In contrast, other participants adopted a more selective strategy, attempting to locate the relevant anti-pattern as quickly as possible using search mechanisms. These students frequently relied on the catalog's search field or even the browser search functionality to directly locate relevant terms.

Our observations also suggest several design improvements that architectural support tools should consider to accommodate these different interaction styles better. For users who rely on comprehensive browsing, clear anti-pattern names and informative summaries directly on the catalog home page can facilitate quick scanning and comparison of multiple candidates. Providing richer previews, such as expandable summaries or modal descriptions, can further support this browsing-oriented process without requiring users to open detailed pages repeatedly. For users who prefer more direct search strategies, more robust search capabilities could improve the interaction experience. Features such as semantic search, similarity-based matching, or AI-assisted recommendations could help developers quickly identify relevant anti-patterns based on the architectural context or problem description. Finally, well-defined, intuitive categories remain essential for supporting structured filtering strategies. Expanding categorization dimensions, such as technical, organizational, or development lifecycle-related perspectives, may further help users narrow the search space before analyzing candidate anti-patterns.

The artifact use differences suggest that architectural support tools should accommodate multiple interaction styles. While some users benefit from browsing-oriented interfaces that facilitate exploration, others prefer more direct and efficient mechanisms for locating information. Supporting both strategies is therefore essential to ensure that such tools remain accessible and useful to a broader range of users. By considering different interaction styles and information-processing strategies, researchers can design artifacts that better support architectural learning and decision-making.

11.2 Implications for Researchers

Our results highlight the importance of empirically investigating how developers interact with architectural knowledge artifacts. In this study, the combination of qualitative feedback and interaction logs revealed different strategies participants used to search for architectural problems in the catalog, including browsing multiple anti-patterns, performing keyword searches, and narrowing the search space with category filters. These results suggest that understanding how users navigate and search within such artifacts can reveal important insights about their usability and effectiveness.

By examining students' interaction patterns together with participants' feedback, we were able to identify opportunities for improving how architectural knowledge is organized and accessed. These findings indicate that studying real user interactions can provide valuable evidence for evolving architectural knowledge artifacts over time. Therefore, researchers designing architectural support tools may benefit from complementing traditional artifact evaluations with analyses of user behavior. Such analyses can help reveal how users actually explore architectural knowledge artifacts and support the iterative refinement of these tools based on empirical evidence.

12 Lessons Learned

Importance of real-world examples – Students reported greater learning gains when exposed to real-world examples presented by the two supporting materials. Feedback indicated that the catalog could even have more examples to support architectural decision-making better. This underscores the limitations of relying solely on textbook-based or overly didactic examples, which often fail to reflect the complexity and nuance of real-world scenarios. Therefore, integrating practical examples into architecture courses is essential to deepen students' understanding and prepare them to apply concepts effectively. Given the detail level of each supporting material, the guidelines are more appropriate for students with no prior exposure to MFE, whereas the catalog is better suited for those with foundational knowledge who aim to reinforce their learning through practical application and problem identification.

Teaching MFE focusing on architectural decisions – Our supporting materials allowed us to teach how MFE is implemented in real-world scenarios without requiring students to engage with low-level programming details. Students could effectively learn about MFE without developing a complete MFE-based system. The course focused primarily on architectural design and included a single lab session to provide hands-on exposure to an MFE implementation, helping students better understand MFE in practice. As a result, the supplementary resources enabled students to grasp key architectural concepts more clearly, without being overwhelmed by implementation complexity. A potential approach would be to balance high-level architectural instruction with practical coding exercises, which can aid in consolidating learning. However, this balance must be carefully managed, as coding exercises may detract from architectural understanding, especially for students with limited development experience.

Using specialized tools for supporting daily activities and enhancing learning – Students feedback indicated that the catalog served as a quick, easy, and centralized source of information, making it an effective support tool. It proved particularly valuable during architectural decision-making, with students emphasizing the importance of having easy access to resources that facilitate such processes. Additionally, students suggested that the catalog could include more information on MFE, further highlighting its potential as a reference and a learning resource. Teaching students how to use practical tools that can be seamlessly integrated into their future development workflows significantly enhances their learning experience.

Learning from architectural mistakes reinforces the importance of sound decision-making – Exposing students to possible architectural mistakes helped them recognize the importance of making careful architectural decisions. By analyzing anti-patterns and their consequences, students realized that poor architectural choices can lead to severe and costly problems in software systems. This exposure made students value spending more time reasoning about architectural decisions before implementing them. Students explicitly mentioned that identifying and understanding potential mistakes early is preferable to correcting them later, as fixing architectural problems often requires significantly more effort. Therefore, presenting anti-patterns during architecture education not only helps students understand common pitfalls but also encourages a more reflective and deliberate approach to architectural design.

Anti-patterns introduce a new pedagogical stage in learning architectural styles – Our results suggest that anti-patterns can be used as a dedicated stage in the teaching process of software architecture. Instead of presenting anti-patterns only as complementary material, they can structure a specific learning phase in which students analyze incorrect architectural decisions and their consequences. Based on our observations, we suggest organizing the teaching of architectural styles in progressive stages: (1) introducing the fundamental concepts of the architectural style, (2) analyzing realistic examples to understand how these concepts appear in practice, and (3) studying anti-patterns to expose common mistakes and highlight the consequences of poor architectural decisions. This final stage helps students deepen their understanding of the architectural style and reinforces the importance of making sound architectural decisions.

13 Conclusion

This paper presents an experience report on teaching MFE within an undergraduate software architecture course. We described our teaching approach, assessments, and the lessons learned throughout the teaching process. In addition, we report two controlled experiments. In the first, we compared the two supporting materials while students solved architectural problems: a catalog of MFE anti-patterns and a presentation containing practitioner-provided guidelines. Based on the results and students' feedback, we improved the anti-pattern catalog by introducing a new template, adding visual elements, and expanding it with 15 new anti-patterns identified in the grey literature. We then conducted a second controlled experiment to compare the new version of the catalog with the original one.

Overall, our results show that both supporting materials were equally effective in helping students learn about MFE and supporting architectural decision-making. The practitioner-provided guidelines helped students understand how MFE is applied in real-world scenarios. At the same time, the anti-pattern catalog increased students' perceived learning and supported activities such as identifying architectural problems, searching for solutions, reviewing concepts, and learning from realistic examples of architectural mistakes. The second experiment revealed no statistically significant differences between the two catalog versions, suggesting that both can effectively support learning, with preferences depending on individual reading styles and information needs. These findings highlight the importance of studying how educational tools are used in practice and continuously evolving them based on user feedback. Without understanding who will use such artifacts and how they actually interact with them, it becomes difficult to design tools that are truly useful for supporting architectural reasoning and that can eventually be adopted in real-world practice.

This work contributes to software engineering education by offering educators a teaching case with a validated methodology for introducing MFE in the classroom, supported by well-designed instructional materials and comprehensive assessments. It addresses a critical gap in the literature, as no prior studies have focused explicitly on how to teach MFE. By presenting a structured approach that tackles both practical challenges and real-world solutions associated with MFE architectures, this work bridges the gap between theoretical knowledge and practice, providing actionable insights for educators and learners. Additionally, the second experiment highlights the importance of empirically validating changes in educational artifacts and studying how users interact with them in practice. We show that different artifact structures may equally support learning while addressing different user preferences. The study also provides lessons learned about teaching architecture with anti-patterns, including their role in reinforcing architectural decision-making and structuring a deeper learning stage after students acquire the fundamental MFE concepts.

Future work could explore workshop-based sessions in which student groups simulate distinct teams responsible for specific subdomains and MFEs, fostering collaboration and mirroring real-world architectural decision-making processes. We also plan to evaluate students' long-term knowledge retention after the course. Future work also includes exploring additional improvements to the catalog by adding new features to support different user types and needs, as well as investigating how AI-based assistants can leverage the anti-pattern catalog to support students learning software architecture. Finally, we intend to assess the effectiveness of the catalog with industry practitioners when analyzing real-world architectures.

Declaration of AI Use

We acknowledge the use of Grammarly and ChatGPT 5.2 to improve the text's spelling, grammar, vocabulary, and style. We also used Gemini 2.5 Flash to speed up Python code generation for visualizations and data analysis. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Data Availability

All instruments, raw data, and detailed results referenced in this paper are publicly available and can be verified at `https://figshare.com/s/7fd92ace78cbb80a024b`.

Acknowledgments

We thank all the participants in the empirical study and USES Research Group members for their support. We also thank Márcio Ribeiro (Federal University of Alagoas) and Igor Steinmacher (Northern Arizona University) for their valuable contributions and insights that helped improve and evolve the anti-pattern catalog. The present work is supported by: Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES-PROEX) – Financing Code 001; CNPq processes 314797/2023-8, 443934/2023-1, and 445029/2024-2; Amazonas State Research Support Foundation – FAPEAM – through POSGRAD 25-26; and Project No. 017/2024 – DIVULGA CT&I/FAPEAM.

References

Anderson, A., Guevara, J. N., Moussaoui, F., Li, T., Vorvoreanu, M., and Burnett, M. (2024). Measuring user experience inclusivity in human-ai interaction via five user problem-solving styles. *ACM Transactions on Interactive Intelligent Systems*, 14(3):1–90.

Anks, D. (2023). Mastering micro frontends: Best practices, pitfalls to avoid, tools and scaling strategies. Available at: `https://dev.to/dr_anks/micro-frontends-dos-donts-tools-and-scaling\-strategies-3n4o`. Online; accessed November 14, 2024.

Antunes, F., Lima, M. J. D., Araújo, M. A. P., Taibi, D., and Kalinowski, M. (2024). Investigating benefits and limitations of migrating to a micro-frontends architecture. *arXiv preprint arXiv:2407.15829*.

Aplyca (2024). Best practices for micro frontends. Available at: `https://www.aplyca.com/blog/best-practices-for-micro-frontends`. Online; accessed November 14, 2024.

Bærbak Christensen, H. (2022). Teaching microservice architecture using devops—an experience report. In *European Conference on Software Architecture*, pages 117–130. Springer.

Brada, P. and Picha, P. (2019). Software process antipatterns catalogue. In *Proceedings of the 24th European Conference on Pattern Languages of Programs*, pages 1–10.

Brown, S. (2023). *The C4 Model for Visualising Software Architecture*. Leanpub.

Brown, W. H., Malveau, R. C., McCormick, H. W. S., and Mowbray, T. J. (1998). *AntiPatterns: refactoring software, architectures, and projects in crisis*. John Wiley & Sons, Inc.

Capdepon, Q., Hlad, N., Seriai, A.-D., and Derras, M. (2023). Migration process from monolithic to micro frontend architecture in mobile applications. In *Proceeding of the International Workshop on Smalltalk Technologies*.

Casas, R. (2020). Rules of micro-frontends. Available at: `https://www.infoxicator.com/rules-of-micro-frontends`. Online; accessed March 17, 2025.

Corbin, J. and Strauss, A. (2014). *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications.

Cordeiro, R., Rosa, T., Goldman, A., and Guerra, E. (2019). Teaching complex systems based on microservices. *GROUP*, 1:1.

Cunningham, W. (2013). Anti patterns catalog. Available at: `http://wiki.c2.com/?AntiPatternsCatalog`. Online; accessed April 25, 2025.

do Nascimento Oliveira, B. R., Rodríguez, L. M. G., Santos, D. S., Galster, M., and Nakagawa, E. Y. (2025). Real-world cases in software architecture education: An experience report. *IEEE Transactions on Education*, 69(1):1–9.

Galster, M. and Angelov, S. (2016). What makes teaching software architecture difficult? In *Proceedings of the 38th International Conference on Software Engineering Companion*, pages 356–359.

Geers, M. (2020). *Micro Frontends in Action*. Simon and Schuster.

Gkamperlo, N. (2020). Micro-frontend "blackbox pattern". Available at: `https://medium.com/@ngkamperlo/micro-frontend-blackbox-pattern-295c40b681e4`. Online; accessed March 17, 2025.

Kaushik, N., Kumar, H., and Raj, V. (2024). Micro frontend based performance improvement and prediction for microservices using machine learning. *Journal of Grid Computing*, 22(2):1–26.

Kazman, R., Cai, Y., Godfrey, M. W., Pautasso, C., and Liu, A. (2023). A better way to teach software architecture. In *Software Architecture: Research Roadmaps from the Community*, pages 101–110. Springer.

Kofler, J. (2020). Como os microfrontends podem ajudar a focar nas necessidades de negócios. Available at: `https://www.infoq.com/br/articles/microfrontends-business-needs/`. Online; accessed November 14, 2024.

Lago, P. and Van Vliet, H. (2005). Teaching a course on software architecture. In *18th Conference on Software Engineering Education & Training (CSEET'05)*, pages 35–42. IEEE.

Laitenberger, O. and Dreyer, H. M. (1998). Evaluating the usefulness and the ease of use of a web-based inspection data collection tool. In *Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No. 98TB100262)*, pages 122–132. IEEE.

Lange, M., Koschel, A., and Hausotter, A. (2019). Microservices in higher education. In *International Conference on Microservices*.

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of psychology*.

Männistö, J., Tuovinen, A.-P., and Raatikainen, M. (2023). Experiences on a frameworkless micro-frontend architecture in a small organization. In *2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C)*, pages 61–67. IEEE.

Mannisto, T., Savolainen, J., and Myllarniemi, V. (2008). Teaching software architecture design. In *Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008)*, pages 117–124. IEEE.

Matteo Figus, Alexander Guensche, H. H. and Mezzalira, L. (2024). Understanding and implementing microfrontends on aws - aws prescriptive guidance. Available at: `https://docs.aws.amazon.com/pdfs/prescriptive-guidance/latest/micro-frontends-aws/micro-frontends-aws.pdf`. Online; accessed March 17, 2025.

Meireles, M. A. C., Rocha, S., Maldonado, J. C., and Conte, T. (2024). An experience report on the use of active learning in empirical software engineering education: Understanding the pros and cons from the student's perspective. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training*, pages 380–390.

Mezzalira, L. (2020). Micro frontends anti-patterns. Available at: `https://www.infoq.com/presentations/microfrontend-antipattern/`. Online; accessed March 17, 2025.

Mezzalira, L. (2021a). *Building Micro-Frontends*. O'Reilly Media, Inc.

Mezzalira, L. (2021b). Chapter 4. discovering micro-frontend architectures. Available at: `https://www.oreilly.com/library/view/building-micro-frontends/9781492082989/ch04.html`. Online; accessed March 17, 2025.

Mezzalira, L. (2021c). Techlead journal: #47 - micro-frontends and the socio-technical aspect. Available at: `https://techleadjournal.dev/page/16/`. Online; accessed March 17, 2025.

Mezzalira, L. (2024). Micro-frontends anti-patterns by luca mezzalira. Available at: `https://www.youtube.com/watch?v=3jygY3LGTKc&ab_channel=Apiumhub`. Online; accessed March 17, 2025.

Moraes, F., Campos, G., Almeida, N., and Affonso, F. (2024). Micro frontend-based development: Concepts, motivations, implementation principles, and an experience report. In *Proceedings of the 26th International Conference on Enterprise Information Systems*, volume 2, pages 175–184.

Newman, S. (2021). *Building microservices*. O'Reilly Media, Inc.

Oliveira, B. R., Garcés, L., Lyra, K. T., Santos, D. S., Isotani, S., and Nakagawa, E. Y. (2022). An overview of software architecture education. In *Congresso Ibero-Americano em Engenharia de Software (CIbSE)*, pages 76–90. SBC.

Peltonen, S., Mezzalira, L., and Taibi, D. (2021). Motivations, benefits, and issues for adopting micro-frontends: a multivocal literature review. *Information and Software Technology*, 136:106571.

Perlin, R., Ebling, D., Maran, V., Descovi, G., and Machado, A. (2023). An approach to follow microservices principles in frontend. In *2023 IEEE 17th International Conference on Application of Information and Communication Technologies (AICT)*, pages 1–6. IEEE.

Pölöskei, I. and Bub, U. (2021). Enterprise-level migration to micro frontends in a multi-vendor environment. *Acta Polytechnica Hungarica*, 18(8):7–25.

Raimundo, J. L. P. (2023). Compositional qualities of microfrontends: The ldod archive. Available at: `https://fenix.tecnico.ulisboa.pt/downloadFile/281870113706102/49372-joao-raimundo.pdf`. Online; accessed March 17, 2025.

Rappl, F. (2024). Top 10 micro frontend anti-patterns. Available at: `https://dev.to/florianrappl/top-10-micro-frontend-anti-patterns-3809`.

Online; accessed March 17, 2025.

Razavian, M., Paech, B., and Tang, A. (2019). Empirical research for software architecture decision making: An analysis. *Journal of Systems and Software*, 149:360–381.

Richardson, C. (2018). *Microservices patterns: with examples in Java*. Simon and Schuster.

Shinde, S. (2022). 4 micro-frontend anti-patterns. Available at: `https://levelup.gitconnected.com/four-micro-frontend-anti-patterns-58aaa9fe1\9d5`. Online; accessed March 17, 2025.

Shukla, V. (2023). A comprehensive guide to micro frontend architecture. Available at: `https://medium.com/appfoster/a-comprehensive-guide-to-micro-frontend-architecture-cc0e31e0c053`. Online; accessed November 04, 2024.

Silva, M. R. d. (2024). Arquitetura reativa cognitiva baseada em microsserviços e micro-frontends para melhorar a experiência do usuário em aplicações bancárias por meio de interfaces adaptativas.

Silva, N., Rodrigues, E., and Conte, T. (2025a). A catalog of micro frontends anti–patterns. In *IEEE/ACM International Conference on Software Engineering (ICSE)*.

Silva, N., Rodrigues, E., and Conte, T. (2025b). Evaluating strategies for teaching micro frontends: Do anti-patterns help? In *Simpósio Brasileiro de Engenharia de Software (SBES)*, pages 522–532. SBC.

Single-spa (2016). single-spa: A javascript router for front-end microservices.

Taibi, D. and Mezzalira, L. (2022). Micro-frontends: Principles, implementations, and pitfalls. *ACM SIGSOFT Software Engineering Notes*, 47(4):25–29.

ThoughtWorks (2016). Micro frontends. *ThoughtWorks Technology Radar*.

Valente, M. T. (2020). Engenharia de software moderna. *Princípios e Práticas para Desenvolvimento de Software com Produtividade*, 1(24).

Vegas, S., Apa, C., and Juristo, N. (2015). Crossover designs in software engineering experiments: Benefits and perils. *IEEE Transactions on Software Engineering*, 42(2):120–135.

Venkatesh, V. and Bala, H. (2008). Technology acceptance model 3 and a research agenda on interventions. *Decision sciences*, 39(2):273–315.

Wessels, B. (2020). Micro front-end architecture at enterprise scale. Available at: `https://medium.com/swlh/micro-front-end-architecture-at-enterprise\-scale-updated-july-2020-9159a4e0cc49`. Online; accessed March 17, 2025.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.