Evaluating Strategies for Teaching Micro Frontends: Do Anti-patterns Help?

Nabson Silva
Institute of Computing
Federal University of Amazonas
Manaus, AM, Brazil
nabson.paiva@icomp.ufam.edu.br

Eriky Rodrigues
Institute of Computing
Federal University of Amazonas
Manaus, AM, Brazil
eriky.rodrigues@icomp.ufam.edu.br

Tayana Conte
Institute of Computing
Federal University of Amazonas
Manaus, AM, Brazil
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 a controlled experiment, 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. **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. **Conclusion**: This paper provides insights into incorporating MFE into a software architecture course, proposes two supporting materials that educators can use to teach MFE, offers practical recommendations for effective instruction, and highlights the potential of the MFE anti-patterns catalog as a valuable learning tool.

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 [22, 25]. This approach facilitates independent testing, development, and deployment of frontend components, enabling teams to work autonomously and reducing the development time for new features [11]. However, some issues faced when adopting MFEs are increased payload size, UX consistency, complex monitoring and debugging, state management, and duplicated code [25]. Many companies, such as SAP, Springer, Zalando, NewRelic, Ikea, Starbucks, and DAZN, have successfully implemented the MFE architecture [33], 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 [2, 7, 12, 19, 23, 26, 27]. 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 [31]. 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 sessions, 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 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 [34] 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 [22, 25]. Geers [11] 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 [22, 33, 34]. 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 [11, 25].

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 [11]. There are three approaches to composing MFEs: Server-side, Edge-side, and Client-side [11, 23, 25, 33]. Communication defines how the screen and fragments interact to deliver a seamless user experience [33]. According to Geers [11], 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 [33].

2.2 Related work

Previous studies have identified several challenges in teaching software architecture. According to Galster and Angelov [10], 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. [13] 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 [15] 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. [20] 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.

To the best of our knowledge, no papers specifically discuss MFE education, and only a few report on experiences teaching microservices. Bærbak Christensen [4] 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. [17] 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. [9] 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 [28]. 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 [6]. Understanding both patterns and anti-patterns equips developers with the knowledge to design robust architectures while avoiding the recurring issues highlighted by these concepts.

In our previous research [31], 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 [1] 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.

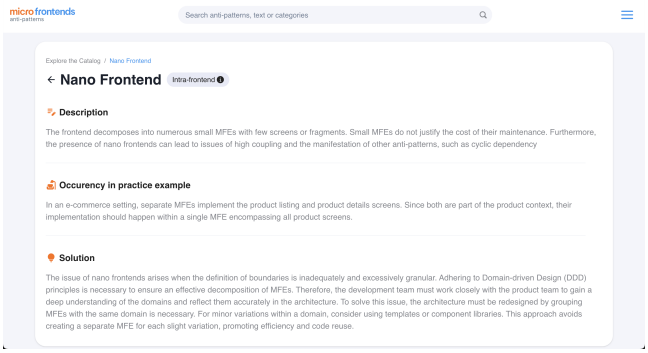

Figure 1: Nano Frontend anti-pattern description from the catalog's web application.

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. [2], Moraes et al. [23], and Silva [30]. 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 [30]. To conclude, we presented a summary of MFE guidelines published by Taibi and Mezzalira [33], Aplyca [3], Kofler [14], Anks [1], and Shukla [29], consolidating recommended practices and solutions.

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

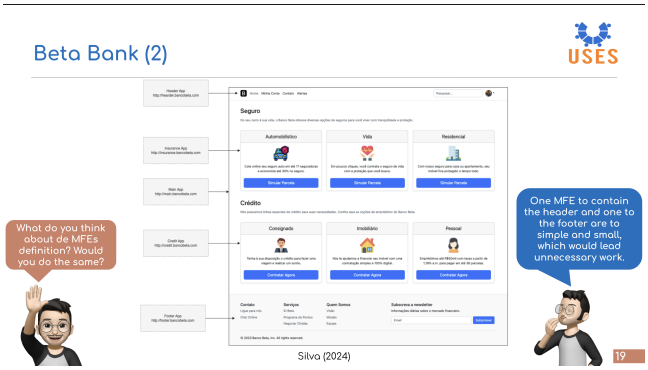

Figure 2: Slide from the practitioner-provided guidelines presentation with an example of architecture and the question we asked to students and its answer.

4 Teaching Approach

This section describes the procedures for teaching MFE to undergraduate Computer Science students at the Federal University of Amazonas (UFAM).

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 [35], and about architecture visualization with C4 Model [5].

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 [24]. Then, we presented the concept of microservices and their key principles. To illustrate some architectural examples, we discussed the following patterns: Remote Procedure Invocation, Asynchronous Messaging, API Gateway, Backend for Frontend (BFF), and API Composition [28].

In the second lecture session, we introduced the concept, benefits, and challenges of MFE primarily based on Geers [11] and Peltonen

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.

et al. [25]. 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 [2], which we presented in the previous session. In the lab session, we explained how the application uses the Single-SPA framework [32] 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 [1, 3, 14, 29, 33] and a catalog of 12 MFE anti-patterns [31], 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 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:

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

(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 [10, 13, 20]. 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 Study Design

To address this study's research questions (Section 5.1), we designed a controlled experiment following the guidelines proposed by Wohlin et al. [38]. The following subsections provide a detailed description of each aspect of the study.

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 Research Questions (RQ):

RQ1

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

To address RQ1, 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.

RQ2

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

For RQ2, we asked students to rate their perceived learning about MFE before and after engaging with the catalog and then compared the two sets of responses.

RQ3

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

For RQ3, 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) [37]. 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 [8].

5.2 Planning

We planned the experiment according to Wohlin et al. [38].

5.2.1 Context Selection. We conducted the experiment with undergraduate students learning about MFE for the first time 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 [38] and evaluating whether the catalog positively influenced students' learning perception, following the approach of Meireles et al. [21].

5.2.3 Hypothesis formulation. We formulated two hypotheses, each corresponding to RQ1 and RQ2. The null hypothesis H_0, related to RQ1, 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 RQ2, states that there is no significant difference in students' perceived learning before and after interacting with the MFE anti-patterns catalog. As RQ3 is addressed through qualitative methods, we did not define a hypothesis for it.

5.2.4 Selection of subjects. Participant selection was based on convenience sampling [38], 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 Experiment 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 [36] 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 [38]. 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.

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 3 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. A total of 23 students participated in the experiment, 12 in Group A and 11 in Group B.

Figure 3: Sequence of steps of the experiment execution.

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 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 (check the ARTIFACT AVAILABILITY Section).

We began by examining boxplots to visually compare the samples within each group. Next, we applied the Shapiro–Wilk test [38] to determine whether the samples followed a normal distribution. Based on the results, we selected appropriate paired statistical tests: the Paired t-Student Test [38] for normally distributed samples, and the Wilcoxon Signed Rank Test [38] for non-normal samples.

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.

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) [37]. TAM assesses users' perceptions of a technology's usefulness and ease of use, which are the two primary factors influencing technology acceptance behavior [16]. We measured the constructs of perceived usefulness, perceived ease of use, and behavioral intention using a 5-point Likert scale [18], 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 [8]. 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.

6 Results

This section presents the results for each RQ outlined in Section 5.1. For RQ1 and RQ2, we analyze the quantitative data collected during the experiment using boxplots and statistical tests to compare the samples. For RQ3, 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 RQ1, we evaluated the students' scores from the first and second assessments. Figure 4 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.

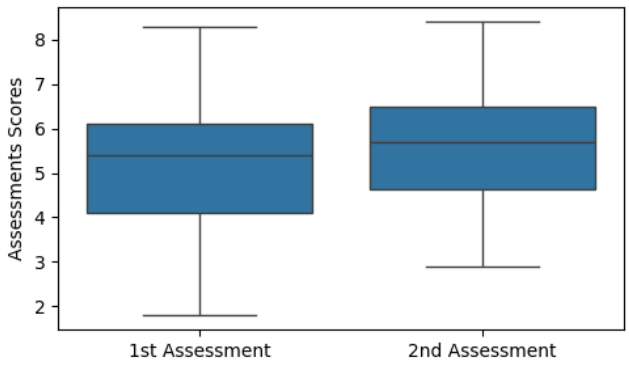

Figure 4: Boxplots presenting the data distribution on the assessment samples.

We selected the Paired t-Student Test for the sample comparison, a parametric test for dependent and normally distributed samples [38]. 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.

RQ1 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 RQ2, we evaluated the students' self-assessed perceived learning before and after engaging with the catalog. Figure 5 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.

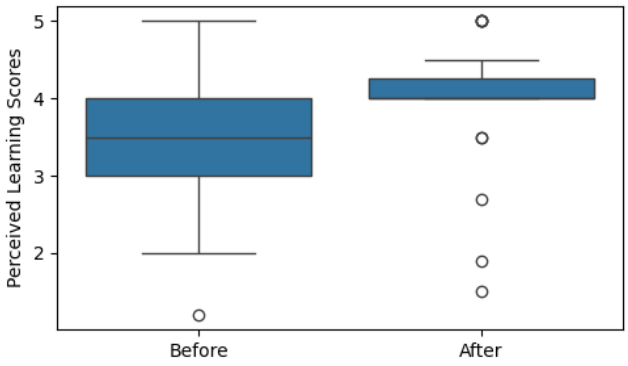

Figure 5: Boxplots presenting the data distribution on the perceived learning before and after engaging with the MFE anti-patterns catalog.

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 [38]. 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.

RQ2 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 RQ3, we asked students to answer a 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.

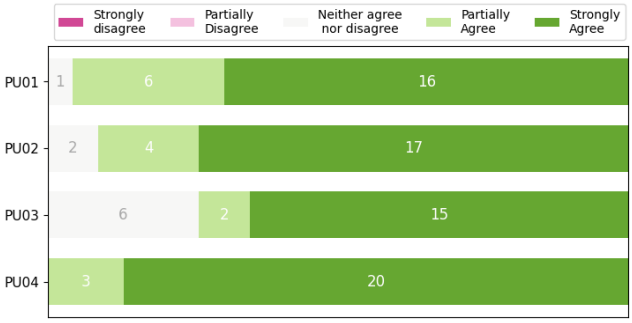

Figure 6: Responses for each sentence in the Perceived Usefulness construct.

6.3.1 TAM and learning statements analysis. Figure 6 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.

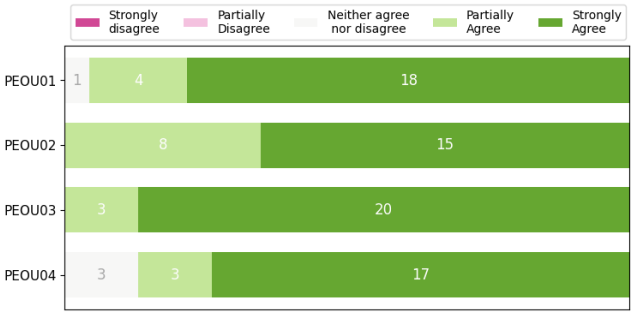

Figure 7: Responses for each sentence in the Perceived Ease Of Use construct.

Figure 7 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.

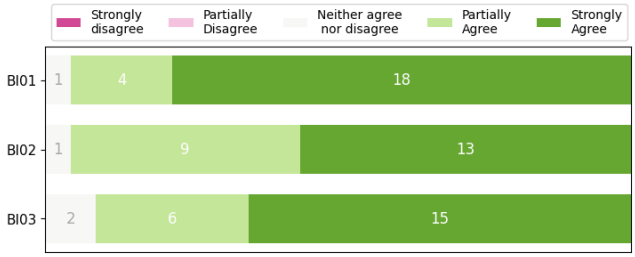

Figure 8: Responses for each sentence in the Behavioral Intention construct.

Figure 8 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 9 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.

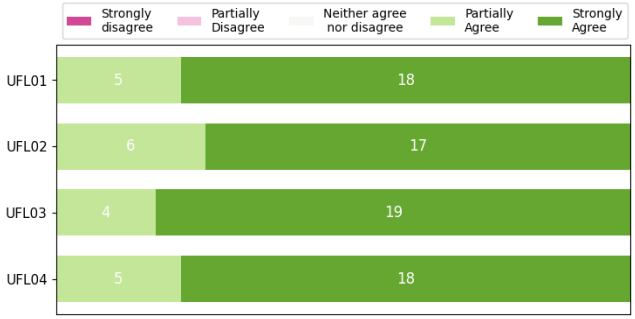

Figure 9: Responses for each sentence related to the catalog's utility for learning.

6.3.2 Qualitative feedback analysis. We analyzed students' qualitative feedback using ATLAS.ti[3], applying open and axial coding [8] 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 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.

RQ3 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 diverse learning strategies.

7 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. [38]:

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.

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

[3] https://atlasti.com/

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.

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.

8 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.

9 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 conducted a controlled experiment to compare the effectiveness of two supporting materials we developed: a catalog of MFE anti-patterns and a presentation containing practitioner-provided guidelines. To further explore the educational value of anti-patterns, we examined whether the catalog improved students' perceived learning and analyzed how students used and perceived its usefulness during one assessment.

Statistical tests indicated that both supporting materials were equally effective in helping students learn about MFE and supporting decision-making. The practitioner-provided guidelines supported students in understanding how MFE is applied in real-world scenarios, offering a practical and comprehensive perspective beyond traditional textbook content. Access to the MFE anti-patterns catalog significantly increased students' perceived learning. Students also reported positive experiences using the catalog, emphasizing its usefulness in identifying problems and solutions, conducting efficient searches for architectural issues, and reviewing core MFE concepts. Overall, the feedback suggests that engaging with tools grounded in real-world architectural challenges can effectively prepare in-training software engineers to address practical issues in MFE development.

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 industry practice, providing actionable insights for educators and learners.

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. Finally, we intend to assess the effectiveness of the catalog with industry practitioners when analyzing real-world architectures.

ARTIFACT AVAILABILITY

All instruments, raw data, and detailed results referenced in this paper are publicly available and can be verified at https://zenodo.org/records/16394584.

ACKNOWLEDGMENTS

We thank all the participants in the empirical study and USES Research Group members for their support. 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

[1] Dr. Anks. 2023. Mastering Micro Frontends: Best Practices, Pitfalls to Avoid, Tools and Scaling Strategies. https://dev.to/dr_anks/micro-frontends-dos-donts-tools-and-scaling-strategies-3n4o. *DEV Community* (3 nov 2023). Accessed: 2024-11-04.

[2] Fabio Antunes, Maria Julia Dias Lima, Marco Antônio Pereira Araújo, Davide Taibi, and Marcos Kalinowski. 2024. Investigating Benefits and Limitations of Migrating to a Micro-Frontends Architecture. *arXiv preprint arXiv:2407.15829* (2024).

[3] Aplyca. 2024. Best Practices for Micro Frontends. https://www.aplyca.com/blog/best-practices-for-micro-frontends Accessed: 2024-11-04.

[4] Henrik Bærbak Christensen. 2022. Teaching microservice architecture using devops—an experience report. In *European Conference on Software Architecture*. Springer, 117–130.

[5] Simon Brown. 2023. *The C4 Model for Visualising Software Architecture*. Leanpub.

[6] William H Brown, Raphael C Malveau, Hays W" Skip" McCormick, and Thomas J Mowbray. 1998. *AntiPatterns: refactoring software, architectures, and projects in crisis*. John Wiley & Sons, Inc.

[7] Quentin Capdepon, Nicolas Hlad, Abdelhak-Djamel Seriai, and Mustapha Derras. 2023. Migration Process from Monolithic to Micro Frontend Architecture in Mobile Applications.. In *Proceeding of the International Workshop on Smalltalk Technologies*.

[8] Juliet Corbin and Anselm Strauss. 2014. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications.

[9] Renato Cordeiro, Thatiane Rosa, Alfredo Goldman, and Eduardo Guerra. 2019. Teaching complex systems based on microservices. *GROUP* 1 (2019), 1.

[10] Matthias Galster and Samuil Angelov. 2016. What makes teaching software architecture difficult?. In *Proceedings of the 38th International Conference on Software Engineering Companion*. 356–359.

[11] Michael Geers. 2020. *Micro Frontends in Action*. Simon and Schuster.

[12] Neha Kaushik, Harish Kumar, and Vinay Raj. 2024. Micro Frontend Based Performance Improvement and Prediction for Microservices Using Machine Learning. *Journal of Grid Computing* 22, 2 (2024), 1–26.

[13] Rick Kazman, Yuanfang Cai, Michael W Godfrey, Cesare Pautasso, and Anna Liu. 2023. A Better Way to Teach Software Architecture. In *Software Architecture: Research Roadmaps from the Community*. Springer, 101–110.

[14] Jürgen Kofler. 2020. Como os Microfrontends podem ajudar a focar nas necessidades de negócios. https://www.infoq.com/br/articles/microfrontends-business-needs/. *InfoQ Brasil* (2020). Accessed: 2024-11-03.

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

[16] Oliver Laitenberger and Horst M Dreyer. 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)*. IEEE, 122–132.

[17] Moritz Lange, Arne Koschel, and Andreas Hausotter. 2019. Microservices in higher education. In *International Conference on Microservices*.

[18] Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of psychology* (1932).

[19] Jouni Männistö, Antti-Pekka Tuovinen, and Mikko Raatikainen. 2023. Experiences on a frameworkless micro-frontend architecture in a small organization. In *2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 61–67.

[20] Tomi Mannisto, Juha Savolainen, and Varvana Myllarniemi. 2008. Teaching software architecture design. In *Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008)*. IEEE, 117–124.

[21] Maria Alcimar Costa Meireles, Sabrina Rocha, Jose Carlos Maldonado, and Tayana Conte. 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*. 380–390.

[22] Luca Mezzalira. 2021. *Building Micro-Frontends*. O'Reilly Media, Inc.

[23] Fernando Moraes, Gabriel Campos, Nathalia Almeida, and Frank Affonso. 2024. Micro Frontend-Based Development: Concepts, Motivations, Implementation Principles, and an Experience Report. In *Proceedings of the 26th International Conference on Enterprise Information Systems*, Vol. 2. 175–184.

[24] Sam Newman. 2021. *Building microservices*. O'Reilly Media, Inc.

[25] Severi Peltonen, Luca Mezzalira, and Davide Taibi. 2021. Motivations, benefits, and issues for adopting micro-frontends: a multivocal literature review. *Information and Software Technology* 136 (2021), 106571.

[26] Rodrigo Perlin, Denilson Ebling, Vinícius Maran, Glenio Descovi, and Alencar Machado. 2023. An Approach to Follow Microservices Principles in Frontend. In *2023 IEEE 17th International Conference on Application of Information and Communication Technologies (AICT)*. IEEE, 1–6.

[27] István Pölöskei and Udo Bub. 2021. Enterprise-level migration to micro frontends in a multi-vendor environment. *Acta Polytechnica Hungarica* 18, 8 (2021), 7–25.

[28] Chris Richardson. 2018. *Microservices patterns: with examples in Java*. Simon and Schuster.

[29] Vivek Shukla. 2023. A comprehensive guide to micro frontend architecture. https://medium.com/appfoster/a-comprehensive-guide-to-micro-frontend-architecture-cc0e31e0c053. *Medium* (13 jul 2023). Accessed: 2024-11-04.

[30] Michael Rodrigues da Silva. 2024. Arquitetura reativa cognitiva baseada em microsserviços e micro-frontends para melhorar a experiência do usuário em aplicações bancárias por meio de interfaces adaptativas. (2024).

[31] Nabson Silva, Eriky Rodrigues, and Tayana Conte. 2025. A Catalog of Micro Frontends Anti–patterns. In *IEEE/ACM International Conference on Software Engineering (ICSE)*.

[32] Single-spa. 2016. single-spa: A javascript router for front-end microservices. https://single-spa.js.org

[33] Davide Taibi and Luca Mezzalira. 2022. Micro-frontends: Principles, implementations, and pitfalls. *ACM SIGSOFT Software Engineering Notes* 47, 4 (2022), 25–29.

[34] ThoughtWorks. 2016. Micro Frontends. *ThoughtWorks Technology Radar* (2016). https://www.thoughtworks.com/pt-br/radar/techniques/micro-frontends

[35] Marco Tulio Valente. 2020. Engenharia de software moderna. *Princípios e Práticas para Desenvolvimento de Software com Produtividade* 1, 24 (2020).

[36] Sira Vegas, Cecilia Apa, and Natalia Juristo. 2015. Crossover designs in software engineering experiments: Benefits and perils. *IEEE Transactions on Software Engineering* 42, 2 (2015), 120–135.

[37] Viswanath Venkatesh and Hillol Bala. 2008. Technology acceptance model 3 and a research agenda on interventions. *Decision sciences* 39, 2 (2008), 273–315.

[38] Claes Wohlin, Per Runeson, Martin H¨ost, Magnus C Ohlsson, Bj¨orn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science & Business Media.