Real-World Cases in Software Architecture Education: An Experience Report

Brauner Roberto do Nascimento Oliveira, Lina María Garcés Rodríguez, Daniel Soares Santos, Matthias Galster, *Member, IEEE*, and Elisa Yumi Nakagawa

***Abstract*—*Contribution:* The article presents two real-world software architecture cases and reports four experiences using these and other cases in four undergraduate computing courses in two universities, encompassing over 400 students. This article also summarizes the lessons learned from the experiences.**

***Background:* To overcome the challenges and difficulties of teaching and learning software architecture, several approaches have been proposed over the years, including case-based learning, which has already drawn attention from the software architecture community. However, it lacks evidence, a deeper analysis, and a detailed report of the use of software architecture cases in learning activities.**

***Intended Outcome:* The main outcome of this work includes the motivation to adopt case-based learning in software architecture teaching, considering the lessons learned to reproduce the experiences.**

***Application Design:* Real-world cases have been employed in different ways: 1) students handled a given case on their own; 2) instructors presented a given case during class time; and 3) several cases were employed throughout an entire course.**

***Findings:* Feedback from the four experiences showed that cases can be considered valuable for software architecture education by mainly motivating and engaging students and immersing them in more realistic scenarios.**

***Index Terms*—Case-based learning, group assignments, software architecture education.**

I. Introduction

SOFTWARE architectures comprise the main software elements, the relation among them, and their properties [1]. Different responsibilities are assigned to these elements, drastically affecting the system's qualities, such as reliability, performance, maintainability, and security. Hence, carefully designing an architecture that meets system requirements is crucial. As a consequence, teaching software architecture and skills required to design architectures to students of computer science-related programs and professionals working on software development is critical.

Software architecture emerged as a practice area in the 1990s, when it began to be taught [2]. Since then, various studies explored diverse approaches to teach students and train industry practitioners [3], aiming to overcome the difficulties and challenges, as reported by different authors [4], [5], [6], [7], [8], [9], [10], [11]. In summary, these difficulties and challenges relate to various factors that involve both theory and practice, in which the experience and knowledge of learners on software design play a determining role to enable meaningful learning [7]. One of such factors involving the theory, for example, is that there is no single, well-accepted definition for what a "software architecture" is. Instead, different groups and individuals provide their definitions, forming a large pool that instructors can pick from.[1] These different definitions could strongly affect the way a course or training is carried out [7]. Choosing more than one can lead to additional problems such as confusing students [7], [10]. Addressing all of them would require effort to develop instructional material and may not suit short teaching or training interventions.

To overcome the challenges and difficulties, different approaches have been explored, including case studies, project-based learning (PBL), and games, as summarized in [3]. They contribute each to a particular aspect of the teaching and learning process [11].

More recently, *case-based learning* has been also explored [9], [12], [13]. These studies investigated the achievement of different learning objectives and the fostering of soft skills such as those associated with teamwork. However, these studies do not provide a deeper analysis of the students' perception regarding cases, how and which cases were applied, or how students dealt with the learning activities using cases.

The main goal of this article is to present an experience report of teaching software architecture by introducing real-world cases in different ways in four computing courses offered from 2019 to 2021 in two universities. In each course, the *experience* aimed at analyzing different factors affecting or affected by using cases and observed from both students' and instructors' perspectives was recorded. This analysis was based

Received 5 January 2024; revised 31 July 2024; accepted 6 November 2025. Date of publication 12 December 2025; date of current version 19 February 2026. This work was supported in part by the Coordination for the Improvement of Higher Education Personnel (CAPES) under Grant 88882.328820/2019-01, in part by the National Council for Scientific and Technological Development (CNPq) under Grant 313245/2021-5 and Grant 440425/2024-7, in part by the São Paulo Research Foundation (FAPESP) under Grant 2023/00488-5, and in part by the Office of Research and Innovation at University of São Paulo (PRPI-USP) under Grant 22.1.09345.01.2. *(Corresponding author: Brauner Roberto do Nascimento Oliveira.)*

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by the National Research Ethics Committee (CONEP–Brazil).

Brauner Roberto do Nascimento Oliveira, Lina María Garcés Rodríguez, Daniel Soares Santos, and Elisa Yumi Nakagawa are with the University of São Paulo, São Paulo 05508-220, Brazil (e-mail: brauner@alumni.usp.br; linagarces@usp.br; danielss@alumni.usp.br; elisa@icmc.usp.br).

Matthias Galster is with the University of Bamberg, 96047 Bamberg, Germany, and also with the University of Canterbury, Christchurch 8041, New Zealand (e-mail: mgalster@ieee.org).

Digital Object Identifier 10.1109/TE.2025.3633160

[1]The Software Engineering Institute has compiled a list of software architecture definitions, which is available at insights.sei.cmu.edu/library/what-is-your-definition-of-software-architecture

on a mix of data sources, such as anonymous questionnaires, interviews, and instructor observations/notes. As a result, it was possible to observe that the use of cases in different ways had a positive impact on the learning process. Based on quantitative and qualitative feedback, a summary is presented concerning the lessons learned and standing challenges when using software architecture cases.

The remainder of this article is organized as follows. Section III describes the cases used in the experiences. Section IV reports the four experiences, while Section V discusses the results. Section VI presents lessons learned and threats to validity. Finally, the conclusion is presented in Section VII.

II. Related Work

The use of case studies and case-based learning approaches is a common practice in teaching software architecture. In this context, cases can serve as illustrative examples of software architecture concepts [6], inspiring educational activities [14], [15] and games [8], [16] and serving as a starting point for applying PBL approaches [10], [17].

Several studies have investigated case-based approaches for educational purposes; among them, five are particularly relevant to the present research. Van Deursen et al. [9] designed and applied a CBL approach in a master's course. The students worked in teams of four and adopted an open-source project for learning architectural concepts on different cognitive levels. Every week, the students attended a theoretical lecture whose concepts were later applied by each team to their project during the next week. This application consisted of describing the concepts addressed in the classroom as they appear in their project. As the course advanced, the documentation developed for the projects covered more and more concepts. At the end of the course, an open book was compiled, comprising the architectural description of several open-source systems. The results were assessed by means of students' feedback and were positive regarding the use of real-world, open-source projects, fostering collaboration through teamwork and learning software architecture concepts by applying them to those projects.

In [18], to teach a course on software architecture design to professionals (with five or more years of experience in the IT industry), two different approaches were employed and compared: one based on CBL and the other on PBL. In both cases, traditional lectures were used to deliver fundamental concepts, whereas workshops were designed and applied throughout the course following either CBL or PBL. In short, the CBL workshops employed a case in which an architecture had to be designed with the support and guidance of the instructor, who acted as a project stakeholder. In the meantime, PBL workshops consisted of an open-ended problem for which an architectural solution was required, and the instructor provided little guidance to the learners. The results of the comparison of both approaches favored PBL over CBL, considering the feedback given by the learners. The authors ran a paired t-test, which pointed to a statistical difference, but did not present an analysis of effect size or evaluate how both approaches impacted individual learning outcomes. Furthermore, the same authors published another study in 2020 [13] and suggested that the high-level guidance of CBL seems more suitable for graduate students, whereas the low-level guidance of PBL is more appropriate for professionals with industry experience.

The research in [12] demonstrated how three case types (bullet, mini, and descriptive) can be used in lectures, workshops, and assessments to achieve specific learning objectives. Bullet cases were used during a discussion lecture after the introduction of basic concepts in a small-group exercise (two to three learners). Mini cases were employed during workshops when groups of four to five learners discussed the same or different cases to further discuss them during the class. Learners (outside the presenting group) acted as stakeholders and raised their concerns about how the solution presented was suitable for the problem. Finally, descriptive cases (detailing a broader scope case study) were used to assess groups of four to six learners responsible for making architectural decisions and producing a concrete architecture of real-life projects. Their work was presented to the class, and other groups asked questions and gave feedback on the decisions. It is worth highlighting that the formats employed for each case type were proposed and discussed in [19] and [20].

Finally, Ouh et al. [13] used a CBL approach in which case studies represented real-life scenarios, and the students performed analysis and applied concepts taught before. The authors did not detail their approach or which and how the cases were employed, but the students' feedback pointed out a need for more case studies and associated activities.

As observed, in summary, the related works present different ways of using cases for teaching software architecture, often relying on the students' feedback for assessment purposes. Hence, this work differs from the others by focusing on different ways of employing cases (group, student-centered case analysis, case as a part of lecture content, and cases along with a whole course), their impact on the perception of students regarding several aspects, and the lessons learned from each experience, based on the data collected and instructors' observations.

III. Description of Cases

This section describes the two cases employed in the experiences and justifications for selecting them.

A. ArXiv Case

The arXiv case consists of a series of three blog posts by a software architect hired in 2017 to rearchitect the software of arxiv.org, a service intended to work as an archive of research papers. In the series of posts, this architect presented the current state of the software, which had been developed since the early 90s, highlighting the need to transition from a monolithic to a microservices architecture. The main reason behind the transition was the difficulty of maintaining and evolving the software due to its huge codebase and lack of a long-term architectural vision. In fact, the requirements changed over the years due to the increasing popularity of the service. This has drawn the attention of project leaders to quality attributes that could represent a risk in the near future, given the steady increase of users, page loads, and

file downloads. Hence, the project arXiv Next Generation (arXiv-NG) was born to tackle these issues. Along with the series of blog posts, the architect produced an open, available architectural documentation[2] following a mix of approaches, such as arch42[3] and C4,[4] which provided a lot of information addressing architectural concerns. The official website also provides information about the project, including the budget for technical staff (development and operation), access and download statistics, and development road maps.

It was considered an excellent case to be used in the classroom for two reasons. First, it addresses the transition from a monolithic to microservices architecture, a movement considered (sometimes mindlessly) by many software companies to reduce costs and time-to-market, aspects critical for competitiveness. However, this transition involves a lot of effort in decision-making. The second reason was the availability of relevant material about the case. The storytelling aspect, which is often associated with engaging students, is covered in a series of posts from the architect's point of view. The architectural documentation is extensive, covering a wide array of concepts and techniques usually present in learning objectives of most software architecture courses [3], such as business goals, quality attributes, architectural synthesis (decisions and documentation associated with them), architectural styles, patterns, and tactics.

B. Dominator Case

This case refers to a challenging five-year project for the development of a mobile robot capable of setting up many dominoes on the ground that can be later knocked down, triggering a chain reaction (also known as the domino effect). This project had as a main stakeholder Mark Rober, a popular content creator on YouTube (over 27 million subscribers), looking to achieve a record of setting up 100 thousand dominoes in 24 h. The development team consisted of three engineers, who had to overcome a series of hardware and software challenges to build a robust robot system. All information about this case is split into two short YouTube videos (one from Mark Rober's perspective[5] and one from Lily Hevesh's,[6] a professional domino artist and content creator) and in the project documentation made by one of the engineers. The videos introduce the project's context, the main goal, some challenges, some technical aspects, as well as the final system in action. The documentation[7] addresses the design (prototypes, hardware components, and the system's software architecture) and details two software components (robot controller and position control mode), which were more challenging in comparison to other pieces. The source code (mostly in C++ and Python) is available on GitHub.[8] In particular, the architectural documentation consists of a short introduction to the system, an informal (lines and boxes) diagram depicting physical nodes along with software modules/components allocated to them, and a short textual description of each module.

This case represents an excellent learning opportunity to introduce the concept of software architecture. The project's context is well-defined, simple, and small enough for students to easily understand the system requirements and stakeholders' goals and grasp the inherent challenges faced by the engineers. Most of these aspects are captured within the short videos and complemented by the project documentation. The source code also plays a fundamental role for students to understand the concept of software architecture, which is at a high abstraction level [7]. In this case, most architectural elements can be easily mapped to the source code structure (e.g., directories and classes), but there are exceptions that can be used to highlight the gap between actual code and abstractions. Moreover, some code elements are not represented in the architectural diagram nor in the textual description, showing that the engineer opted to hide details considered not relevant to show in a high-level view of the system. Last but not least, the informal architectural representation, while very useful for a general understanding, has its drawbacks. The semantics of the diagram are implicit, leading to different interpretations. For example, different colors are used for the boxes, lines between boxes, or boxes represented inside others without describing their meaning explicitly. When faced by students, these problems can motivate the adoption of formal/semi-formal models to represent software architectures.

IV. Report of Experiences

This section presents the four experiences based on cases for software architecture education, as summarized in Table I. Each experience consists of one or more activities designed and applied in one or two computing-related courses using one or more cases. In total, more than 400 students participated. Following, each experience is detailed and classified as: 1) use of a given case in active learning; 2) presentation of a given case to students by the instructor; and 3) use of several cases in active learning.

A. Experiences #1 and #2

The first two experiences aimed to answer the research question "**RQ1: What is the perception of students when using a given software architecture case in active learning?**" The experiences with the introduction of a real-world case were in undergraduate software engineering courses taught in the first semester of 2019 and 2020. These courses usually include topics of software engineering, such as software development processes, requirements engineering, architecture, and testing. Specifically for software architecture, the topics are software architecture definition, architectural process, software quality, architectural description, views, styles/patterns, and evaluation. An activity based on the arXiv case was conducted, considering the following *learning objectives*: 1) identify factors impacting architectural decisions (or the architecture); 2) explain the reasons behind architectural decisions; 3) apply

[2]arXiv-NG Architecture: https://arxiv.github.io/arxiv-arxitecture/

[3]https://arc42.org

[4]https://c4model.com

[5]Video: World Record Domino Robot (100-k dominoes in 24 h).

[6]Video: HUMAN versus ROBOT Domino Building Machine! (w/ Mark Rober).

[7]Domino Robot—Baucom Robotics.

[8]Github—alexbaucom17/DominoRobot.

TABLE I
SUMMARY OF THE EXPERIENCES

ID	Description	Semester	Modality	Course Name	Cases	# of students	Feedback approach
#1	Use of a given case in active learning	2019-1	In-person	Software Engineering	arXiv	78	Questionnaire
#2		2020-1	Emergency remote teaching	Software Engineering	arXiv	25	Questionnaire
#3	Presentation of a given case to students by instructor	2020-2	Emergency remote teaching	Object-Oriented Analysis and Design	arXiv	166	Questionnaire
#4	Use of several cases in active learning	2021-2	Emergency remote teaching	Software Architecture	Group choice and Domino Robot	141	Interview

the architectural process to reconstruct a software architecture; 4) evaluate an architectural style regarding its suitability to a given problem; and 5) explain how architectural evaluation can be used during an architectural style transition.

The students had already been introduced to basic concepts on software architecture when these experiences took place; hence, these experiences aimed to consolidate the knowledge about such concepts and achieve the learning objectives. In short, the activity consisted of students reading the case material and further answering a series of questions (individually or in duos or trios). Since the case was originally written in English, the case material was translated into Portuguese, leaving students to choose which version they felt more confident with. Throughout the activity, the students could seek additional information on the internet or even ask the instructors. After answering the questions, another questionnaire addressing several aspects related to the research question was anonymously applied to all students. The goal of this questionnaire was to assess the students' perspective on using arXiv and collect information on how they approached the exercise questions. This questionnaire addressed: 1) previous knowledge in that case; 2) time spent to deal with the activity; 3) the approach followed by the students to solve the questions; 4) how interesting was the case; 5) quality of the material; 6) case usefulness for understanding concepts; 7) the link between concepts and real-life facts; and 8) whether they would like to do more activities using real-world cases.

In total, 103 students participated in the activity for around 100 min. The feedback from 78 students (53 in Experience #1 and 25 in Experience #2) was collected using a questionnaire right after the activity.

B. Experience #3

Experience #3 occurred in an undergraduate course on object-oriented analysis and design (online classes during the pandemic), which took place in the second semester of 2020. The course's structure addresses software modeling, employing UML diagrams and design patterns [21]. With this experience, the goal was to answer "**RQ2: What is the perception of students when a given software architecture case is presented by an instructor?**" The main *learning objectives* were: 1) provide examples of architecturally significant requirements; 2) identify architecturally significant requirements in the context of a particular software project; and 3) understand the relevance of software architecture in software development.

In this experience, a case-based lecture was delivered to students, who had to perform an activity individually. The concept of software architecture was introduced (following the structural definition [22]) and explained with the support of a diagram (depicting the client-server model); following, the arXiv case was presented. Throughout the lecture, the focus was on raising awareness about the importance of software architecture, describing the history behind arXiv, the decisions taken that led to the current state of arXiv, and the reasoning and effort behind moving to another architectural style. During the lecture, questions were asked to engage students to participate with their opinions and promote relevant discussions. At the end of the lecture, students answered two questions individually regarding the importance of software architecture and factors influencing its design. Following, another questionnaire was applied to assess the students' perspective on the use of arXiv and their opinion about the interest and usefulness of the case to learn software architecture. The factors analyzed were: 1) previous knowledge of software architecture; 2) previous knowledge about the case; 3) how interesting the case was; 4) case usefulness for understanding concepts; 5) the link between concepts and real-life facts; and 6) whether they would like to do more activities using real-world cases.

C. Experience #4

Experience #4 aimed to answer **"RQ3: What is the students' perception of using several software architecture cases for learning?"** This experience took place in the second semester of 2021, yet during the remote teaching modality. The course on software architecture was offered to 141 undergraduate students of computer science, computer engineering, and information systems.

Differently from the previous experiences when the case(s) and associated material were provided, students could search for and pick real-world cases to solve the exercises. To support them in such a task, certain tips on potential sources of cases were provided, such as the search strategies (e.g., search strings to be used in publication databases) and information to look for within the case material. Since some students were already involved in industry software projects, they were allowed and encouraged to use such cases (if allowed by the company). The decision behind asking students to find cases was based on two assumptions. First, students could pick cases they found interesting and were eager to have a better understanding, thus motivating them in the process and positively impacting learning. Cases brought from companies

TABLE II
SOFTWARE ARCHITECTURE FIRST COURSE STRUCTURE

#	Lecture Topic	Case	Learning Objective
1	SA Definition	Student's choice	**1**. Summarize the context of a software project from a software engineering perspective. **2**. Describe the architecture of a software project in terms of its components and the relationship among them.
2	Relevance of SA	Student's choice	**1**. Identify the relevance of software architecture for software development. **2**. Compare the relevance of software architecture in projects of different sizes. **3**. Understand the relationship between the software architecture of a system and the success of a software project.
3	SA Overview	Student's choice	Particular to each case.
4	Architectural Process	Student's choice	**1**. Describe an architectural method/process/technique used within a project. **2**. Explain how architectural methods/processes/techniques can support software development.
5	Quality Attributes	Student's choice	**1**. Identify relevant quality attributes for a software project. **2**. Explain the importance of particular quality attributes for a software project. **3**. Apply quality attribute scenarios to specify quality requirements.
6	Architectural Synthesis	Domino Robot	**1**. Identify relevant quality attributes for a software project. **2**. Explain the importance of particular quality attributes for a software project. **3**. Understand the link between the context and architectural solutions **4**. Identify which software modules would be affected by changes in a particular module of a software project (apply impact analysis).
7	Architectural Tactics	Domino Robot	**1**. Select tactics to improve a particular quality attribute within a software project. **2**. Analyze the consequences of application of tactics within software projects (trade-off analysis).
8	Architectural Representation	N/A	**1**. Understand architectural views
9	C4 Model	Student's choice	**1**. Apply C4 model to represent the architecture of a software system.
10	Architectural Evaluation	Student's choice	**1**. Design a plan to evaluate the architecture of a system using ATAM.

where they worked could also benefit the students' learning process. Second, all students could benefit from learning about different projects and cases brought by different groups of students. The diversity of cases can support students in understanding how the software architecture is impacted in different ways depending on the project context. It is also useful to see several architectural solutions and technologies, possibly inspiring students in their current or future projects. A time frame was dedicated during some classes exclusively for students to share and discuss their answers to the exercises' questions with other students. Finally, the Domino Robot case was introduced to complement all other cases. Table II summarizes the whole course's structure, indicating for each lecture its topic, the case study employed (whether students selected their own case or used the Domino Robot case), and the corresponding learning objective.

At the end of the course, a semi-structured interview was conducted, containing open-ended and direct questions. Eight students (of all students) accepted the invitation. The main goal of this interview was to gather feedback on the use of cases (chosen by students) and the Domino Robot case (chosen by us), as well as the motivation for using cases for education.

V. RESULTS

This section discusses the results of each experience based on the collected data.

A. Experiences #1 and #2

Fig. 1 summarizes the general information about Experiences #1 and #2. Concerning configurations of student groups, trios were the most frequent choice (59%), followed by pairs (31%), and solo (10%). Regarding the time spent by students to read the material, the same proportion of students took from 30 to 50 min (45%) or over 50 min (45%); only a few took less than 30 min (10%). Moreover, the majority of students had no prior knowledge about arXiv (85%). Throughout the activity, 68% of students stated they had to seek information to better understand the case.

TABLE III
CONTINGENCY TABLE FOR TIME SPENT AND SEEK INFORMATION. FISHER'S EXACT TEST YIELDED A p-VALUE = 0.01

	Seek Additional Information	
Time Spent	**Yes**	**No**
<30 min	2 (25%)	6 (75%)
30-50 min	23 (80%)	12 (20%)
>50 min	28 (66%)	7 (34%)

Fig. 2 summarizes the perception of students on the arXiv case and the exercise. Overall, most students agreed or strongly agreed with the five issues evaluated, i.e., if the case was interesting, if the material was adequate to understand the case, if the case was useful to understand the concepts presented in the course, and useful to connect such concepts to real-world facts, and if students would like to have more exercises and activities involving cases.

Fig. 3 summarizes how students organized themselves to solve the exercise. Most students read the questions before reading the case, while just around one-third of students solved the questions collaboratively or took relevant notes while reading the case material. In more detail, during Experiences #1 and #2, students formed pairs or trios, but they could solve exercises individually as well. This limit on the group size was supposed to encourage them to explore the case material and answer questions in a collaborative way. Too many students would lead to extensive discussions, exceeding the amount of time available for the activity. On the other hand, students working individually could not benefit from teamwork, which has the potential to develop soft skills and possibly provide better responses to the questions. Analyzing only pairs and trios, it was observed that around half of them preferred to solve the questions collaboratively, meaning that around half of them split the questions among group members and used a concurrent approach to finish the exercise.

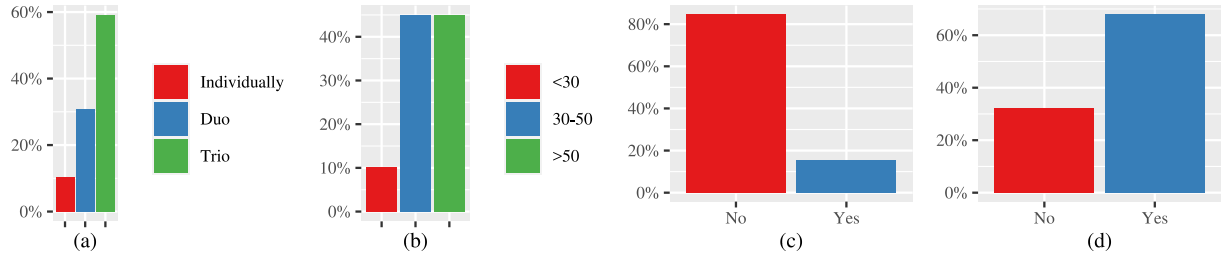

Fig. 1. Barplots for Experiences #1 and #2: (a) group configurations, (b) time spent by students reading and solving the exercise (in minutes), (c) students without previous knowledge about arXiv, and (d) students who had to search for additional information to solve the case exercise.

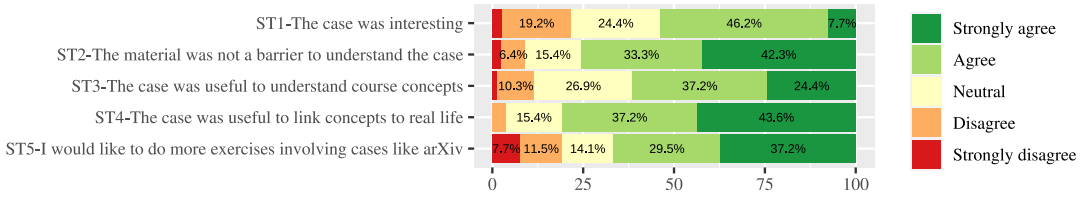

Fig. 2. Experiences #1 and #2: Proportion of statement evaluations regarding the case, case material, and activity (n = 78).

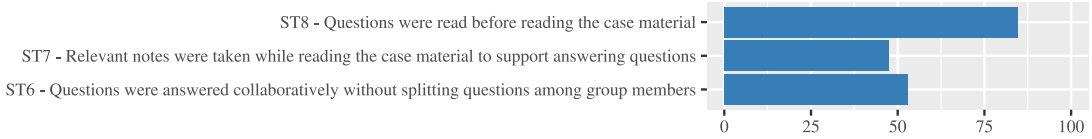

Fig. 3. Experiences #1 and #2: Approaches adopted by students to solve the exercise.

Table III presents the contingency table for the time spent reading the case material and time to seek additional information. In short, it shows how closely correlated the two qualitative variables are. Notice that groups that sought information were also more likely to spend more time (30 min or over) to solve the case, i.e., it was possible to identify a relationship between time spent and seeking material, as tested with Fisher's Exact Test. The time spent reading case material was a surprising result for us, particularly in those cases where they spent over 50 min, which accounts for nearly half the students. One of the most relevant factors affecting this time was the need to seek additional information regarding the case or associated concepts. In fact, 80% of students who took between 30 and 50 min and 66% of those who spent over 50 min seek additional information. Most students (75%) who spent less than 30 min, however, did not look for further information. These results may indicate that experienced students did not have to look for extra information to solve the case, spending less time reading the material. Spending more time during the exercise, however, is not necessarily a negative sign. Students may have spent more time because they were engaged in discussing the case, seeking information, and this may have positively impacted their learning process.

Most students did not know arXiv beforehand. It was expected since it is mostly popular in the research community of certain areas. In fact, most of them (54%) agreed or strongly agreed that the case was interesting (ST1 in Fig. 2). Negative and neutral scores may be related to the type of system addressed or the way the case was described by the original author. While the scores for the statements shown in Fig. 2 are positive in general, some statements scored better than others. ST4 (*"The case was useful to link concepts to real-life"*), for example, had the highest agreement among students. This result reinforces the suitability of real-world cases to bridge theory and practice. On the other hand, ST1 (*"The case was interesting"*) and ST5 (*"I would like to do more exercises involving cases like arXiv"*) had a lower level of agreement when compared to the others. Regarding ST1, students who did not find the case interesting may not have enjoyed the context or application domain. Perhaps this result would be different for cases from other domains or presented in a more engaging way. Regarding ST5, the disagreements may be associated with the effort required to read the case material, seek additional information, and solve the exercise questions.

Answering **RQ1**, from the perspective of most students, cases are well-accepted as a learning approach when they have to actively explore and learn the case by themselves, even when an additional effort is required to understand the case and solve related exercises.

B. Experience #3

Fig. 4 shows the results of Experience #3. The results indicated a positive perception of the use of a given case.

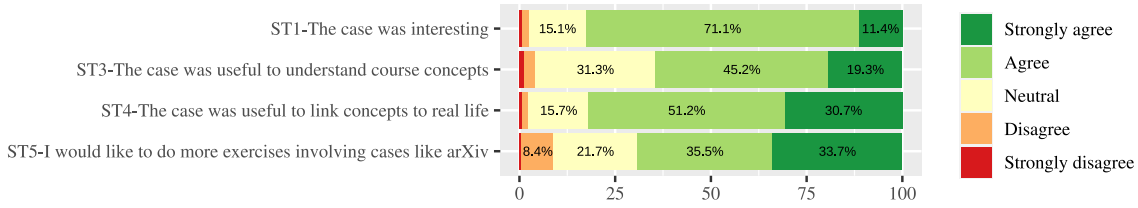

Fig. 4. Experience #3: Proportion of statement evaluations regarding the case, case material, and activity (n = 166).

Overall, this experience yielded a high score (agree and strongly agree) for all statements. There is also a noticeable difference between agreement levels for each statement. For example, both ST4 and ST5 had a higher level of strongly agree responses compared to ST1 and ST3. In other words, the students felt more confident about the usefulness of the case by linking theory to real-life practice, or about doing more exercises than they believed the case would be interesting or useful for when learning course concepts.

Compared to Experiences #1 and #2, responses to ST1 had a much higher level of agreement (agree + strongly agree) in #3, possibly indicating the case is more interesting when presented by the instructor. While a small increase of neutrality for ST3 and ST5 can be observed in #3, students showed less disagreement (disagree + strongly disagree) for ST1, ST3, and ST5. Moreover, Experience #3 also had low strongly agree evaluations for ST4, and slightly low for ST3 and ST5, which possibly indicates that students were not against the usefulness of cases.

Answering **RQ2**, it was observed that the results of Experience #3 reinforce the positive perception students had about the employment of real-world cases for learning purposes, regardless of the way the cases are employed. The differences in scores (comparing Experiences #1, #2, and #3) are possibly due to the way the case was applied; Experience #3 required more attention from the students but less effort to learn the case and solve the exercise.

C. Experience #4

This section reports the results obtained from the analysis of the individual interviews with eight students. Below, the answers for six main questions are summarized: how students selected the cases, the number of cases selected, how students think about the cases selected, how about the domino robot case, and the value of cases for software architecture education.

1) How Did the Group Select the Case(s)? To select cases, the students considered two classes of cases: known cases and nonknown cases. Regarding known cases, one or more group members had previous knowledge about them. These cases referred to industry and/or course software projects in which they had participated or were participating. Concerning nonknown cases, students found them on the Internet or in scientific papers. These cases often addressed popular software projects/systems such as Netflix; in turn, one interviewee highlighted the willingness of Netflix staff to share the experience with architecture.

Different rationales were considered to decide which cases to use. From the interviews, it was identified that familiarity was one of the most frequent factors. In practice, the rationale behind this factor is that the ease with apply concepts from the course can be applied to that case. Another rationale is the possibility of students mapping software architecture concepts to the projects where they are currently working, benefiting both the learning process and the project under development. While these factors were considered more often, others, such as the language in which the case is described, the use of diverse resources to describe the case (e.g., figures, diagrams, and video presentations), and the availability of material, were also mentioned.

2) Were More Than One Case Considered for Selection and Use Throughout the Course? Most interviewees answered that their groups came up with a list of cases that could be used. They also informed that after solving the first exercise with a selected case, the groups often opted to pick different cases to solve other exercises. This change was justified in a great deal by the difficulty of matching the case content to the exercise questions. Another reason to select other cases was to make the questions easier to solve by finding cases that explicitly address what had been asked in the exercise.

3) What Did They Think About the Case(s) Selected? Most interviewees had a positive thought for all cases. In particular, interviewees whose groups selected more than one case were able to compare cases. They argued that the preference for a particular case was due to the architectural complexity and novelty of the case; otherwise, they also affirmed that cases that addressed simple and/or well-known architecture might limit their benefits for learning purposes. Another reason for preferring one case over others was the availability and diversity of materials. Some interviewees also stated that those cases that they had a good previous knowledge of could be more beneficial in the sense of enabling a deeper and straightforward application of the concepts explored during the course.

4) Were the Selected Cases Useful for Learning? The interviewees' perception of the usefulness of the cases for learning purposes was positive in general. For them, the cases were relevant to understand and link the course's contents to real-world projects. In particular, an interviewee mentioned that cases were useful to learn about problems/situations that they could face in the future. Some interviewees considered some cases were less useful than others, mainly due to the simplicity of those cases or the lack of novelty (e.g., a student familiar with client-server architectures may not appreciate cases focused on this architectural style).

Another interviewee suggested that more cases could have been helpful to increase or improve the knowledge even further.

5) How was the Domino Robot Case Explored in the Learning Process? The interviewees reported different aspects of the Domino Robot case. Most recognized the role of the case to support the learning process, as this case deals with different topics of software architecture, e.g., architectural documentation and styles. The availability of several materials addressing different parts of this case was also highlighted as a big pro, particularly for students who had selected cases that had scarce, single-format material on the architecture. On the other hand, some interviewees pointed out a few aspects that negatively impacted their perception of the case. First, some tagged it as a simple case because it employs a client-server style, a short and simple architectural documentation, or both. This perception came from more experienced students (who already knew client-server architectures) or from groups that had previously selected cases with more complex architectures. Another reason was that the Domino Robot architecture description does not include decisions made and their rationale, as also found in other cases found by the students.

In the case of Domino Robot, all documentation was provided, so students could start from whatever made sense for them. All material was provided, and students were recommended to see the video first and further explore the material. During the interviews, it was observed that almost all interviewees viewed the video after reviewing the documentation associated with the Domino Robot. The Domino Robot source code drew some attention but was not explored in depth; interviewees argued that it was not necessary to solve the exercises. An interviewee mentioned that the group first read the documentation rather than the video; however, they had difficulty understanding the concepts, so they switched to the video for clarification.

6) Are Experiences Addressing Cases Valuable for the Software Architecture Education? Most interviewees pointed out that learning using real-world cases was very effective and sometimes better than traditional methods, which are not often centered on the learners, who remain passive for most of the educational experience. At the same time, some of them recognized some challenges for instructors and students. Employing cases usually implies additional effort and time when planning teaching interventions and learning activities [23]. From the students' perspective, they highlighted that the main challenges were: 1) finding and selecting cases with comprehensive material (some cases were interesting but the lack of material made the group move to another one); 2) the terminology employed in the case material, which included jargon and technology the students did not know, requiring extra effort to explore and understand the cases; and 3) the understanding of the inherent and implicit information in the case material. They also informed that they had two previous experiences in other courses using real-world cases, mostly as examples, in which case(s) were used to demonstrate relevant concepts. Hence, the adoption of cases could be explored in other courses, too.

Finally, answering **RQ3**, it was observed that the use of several cases was overall well-accepted, with each case contributing to different aspects of the learning process. It is relevant to highlight that students will probably face challenges when choosing cases to base their learning on.

VI. Discussion

This section presents the main lessons learned, the threats to the validity of this work, and means to mitigate them.

A. Lessons Learned

The main learned lessons of this work are as follows.

1) *Providing Support to Students to Find, Select, and Understand Software Architecture Cases is Necessary:* Despite the guidance provided for the activities to be performed, it was observed that students usually need extra guidance for finding and selecting software architecture cases that can match the exercises. Hence, details on which sources, search terms, and the kind of information should be looked for can support them, while not taking away the opportunity to select the cases they are willing to work on. A list of cases with precurated material can also be a strategy, despite the effort required from instructors to select and organize such material. When it comes to understanding the cases and solving the exercises, students employ different approaches. In some cases, this led to less collaboration among students and more effort to understand the cases. Hence, guidance on how cases could be approached may overcome such a problem.
2) *Terminology Associated With the Cases Can be a Difficulty:* Since some case materials were provided to students in their original form, a number of students did not know some terminology contained in the case description, such as terms, concepts, technologies, or approaches (techniques and methods). Consequently, students spent time on external material to understand those elements before resuming the activities. This might have broken the immersion into the case, besides adding extra effort to understand the case. While new concepts can be useful to trigger students' curiosity, adding a glossary or even presenting an additional explanation of such terminology could benefit the whole activity. Tailoring the case to suppress or adapt such terminology is also viable.
3) *Size of the Cases Should be Considered:* One issue faced when employing the arXiv case was its size. Initially, it was decided not to trim down or summarize it so students could choose between what was relevant or not to solve the exercises. However, the tradeoffs should be considered. Besides increasing the effort required by students, too much extraneous information can hinder the main goals of the activities. Breaking the cases into smaller parts to be addressed over more classes can also be a strategy.
4) *Order and Complexity of Cases Should be Planned:* When designing a course that is supposed to address

several cases, it is essential to analyze their content and compare them before deciding when to introduce each one and how each contributes to the course. Besides introducing more complex cases progressively, attention should be paid to how the next case addresses new knowledge for the students. For instance, a simpler case like the Domino Robot will be more appreciated at the beginning of the experience than arXiv.

B. Threats to Validity

The main threats to the validity of this work are related to how the questionnaires and interviews were employed. Concerning the questionnaires, since this study took place during regular courses, the students' concerns with their final grades could have biased their responses, leading to more positive scores, or even fewer or no negative scores for the statements. To mitigate this threat, students were asked to respond to the questionnaires anonymously. The interest of instructors in their feedback was highlighted, regardless of whether it is positive or negative. Regarding the interviews, all of them were carried out after finishing the course grading process, encouraging students to provide positive and negative feedback comfortably. In this case, however, the period between the course ending and the interviews might have affected the students' capabilities to remember particular details of the whole experience, negatively affecting their feedback.

VII. Conclusion

Real-world cases are well-known, relevant knowledge sources for educational purposes in diverse fields; so this work reports different experiences exploring cases in software architecture education. Based on feedback from more than 400 students, it was observed that software architecture cases are a well-accepted, useful, and effective means, mainly motivating students in the learning process and immersing them in more realistic scenarios. Such cases may also be relevant for teaching nontrivial, complex topics, such as those in software architecture, including the software architecture concept, quality tradeoffs, and architectural decision-making. Another important observation is that cases need to be well-prepared and tailored (in terms of the materials associated with them, taking care to consider their size and complexity) and well-positioned in the course in line with the learning objectives to effectively deliver benefits over other traditional learning methods and, ultimately, impact students' learning.

The main aim of this work was to share the experience with the software architecture community, particularly higher education teachers/instructors, corporate trainers, and educational institutions, about the benefits of adopting real-world cases.

Future research will focus on conducting a detailed analysis, supported by empirical evidence, of students' performance in learning software architecture through real-world case studies. The study will also aim to understand how case-based learning stands out when compared with other active learning approaches, and how it can be effectively combined with complementary methodologies to enhance the teaching of software architecture in undergraduate courses. The overall goal will be to investigate whether case-based learning contributes to a deeper understanding of software architecture concepts. In addition, it is intended to formalize guidelines to support instructional design decisions and the selection and application of case studies aligned with specific learning objectives and software architecture topics.

References

[1] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 4th ed., Reading, MA, USA: Addison-Wesley, 2021.

[2] M. Shaw and D. Garlan, *Software Architecture-Perspectives on an Emerging Discipline*. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

[3] B. R. N. Oliveira, L. Garcés, K. T. Lyra, D. S. Santos, S. Isotani, and E. Y. Nakagawa, "An overview of software architecture education," in *Proc. Anais Do 15th Congresso Ibero-Americano Em Engenharia De Softw. (CIbSE)*, Jun. 2022, pp. 76–90.

[4] P. Lago and H. van Vliet, "Teaching a course on software architecture," in *Proc. 18th Conf. Softw. Eng. Educ. Training (CSEET)*, Apr. 2005, pp. 35–42.

[5] T. Mannisto, J. Savolainen, and V. Myllarniemi, "Teaching software architecture design," in *Proc. 7th Work. IEEE/IFIP Conf. Softw. Archit. (WICSA)*, Feb. 2008, pp. 117–124.

[6] C. R. Rupakheti and S. V. Chenoweth, "Teaching software architecture to undergraduate students: An experience report," in *Proc. IEEE/ACM 37th IEEE Int. Conf. Softw. Eng.*, vol. 2, May 2015, pp. 445–454.

[7] M. Galster and S. Angelov, "What makes teaching software architecture difficult?," in *Proc. IEEE/ACM 38th Int. Conf. Softw. Eng. Companion (ICSE-C)*, May 2016, pp. 356–359.

[8] H. Cervantes, S. Haziyev, O. Hrytsay, and R. Kazman, "Smart decisions: An architectural design game," in *Proc. IEEE/ACM 38th Int. Conf. Softw. Eng. Companion (ICSE-C)*, May 2016, pp. 327–335.

[9] A. Van Deursen et al., "A collaborative approach to teaching software architecture," in *Proc. ACM SIGCSE Tech. Symp. Comput. Sci. Educ.*, Mar. 2017, pp. 591–596.

[10] S. Angelov and P. de Beer, "Designing and applying an approach to software architecting in agile projects in education," *J. Syst. Softw.*, vol. 127, pp. 78–90, May 2017.

[11] P. Lago, J. F. Cai, R. C. de Boer, P. Kruchten, and R. Verdecchia, "DecidArch: Playing cards as software architects," in *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, 2019, pp. 7815–7824.

[12] E. L. Ouh and Y. Irawan, "Applying case-based learning for a postgraduate software architecture course," in *Proc. ACM Conf. Innov. Technol. Comput. Sci. Educ.*, Jul. 2019, pp. 457–463.

[13] E. L. Ouh, B. K. S. Gan, and Y. Irawan, "Did our course design on software architecture meet our student's learning expectations?," in *Proc. IEEE Frontiers Educ. Conf. (FIE)*, Oct. 2020, pp. 1–9.

[14] S. A. Butler, "A client/server case study for software engineering students," in *Proc. 12th Conf. Softw. Eng. Educ. Training*, 1999, pp. 156–165.

[15] H. B. Christensen and A. Corry, "Lectures abandoned: Active learning by active seminars," in *Proc. 17th ACM Annu. Conf. Innov. Technol. Comput. Sci. Educ.*, Jul. 2012, pp. 16–21.

[16] C. H. Montenegro, H. Astudillo, and M. C. Gómez Álvarez, "ATAM-RPG: A role-playing game to teach architecture trade-off analysis method (ATAM)," in *Proc. XLIII Latin Amer. Comput. Conf. (CLEI)*, Montenegro, Sep. 2017, pp. 1–9.

[17] H. Van Vliet, *Software Engineering: Principles and Practice*, vol. 13. Hoboken, NJ, USA: Wiley, 2008.

[18] O. E. Lieh and Y. Irawan, "Teaching adult learners on software architecture design skills," in *Proc. IEEE Frontiers Educ. Conf. (FIE)*, Oct. 2018, pp. 1–9.

[19] C. F. Herreid, "Case studies in science—A novel method of science education," *J. College Sci. Teach.*, vol. 23, no. 4, pp. 221–229, Feb. 1994.

[20] C. F. Herreid, "Case study teaching," *New Directions for Teaching Learn.*, vol. 2011, no. 128, pp. 31–40, 2011.

[21] C. Larman, *Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design and the Unified Process*. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[22] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd ed., Reading, MA, USA: Addison-Wesley, 2003.

[23] O. E. Lieh and Y. Irawan, "Exploring experiential learning model and risk management process for an undergraduate software architecture course," in *Proc. IEEE Frontiers Educ. Conf. (FIE)*, Oct. 2018, pp. 1–9.

Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - ROR identifier: 00x0ma614