

Singapore Management University

## Institutional Knowledge at Singapore Management University

---

Research Collection School Of Computing and Information Systems

School of Computing and Information Systems

---

5-2023

### Overcoming challenges in DevOps education through teaching methods

Samuel FERINO

Marcelo FERNANDES

Elder CIRILO

Lucas AGNEZ

Bruno BATISTA

*See next page for additional authors*

Follow this and additional works at: [https://ink.library.smu.edu.sg/sis\\_research](https://ink.library.smu.edu.sg/sis_research)



Part of the [Instructional Media Design Commons](#), and the [Software Engineering Commons](#)

---

#### Citation

FERINO, Samuel; FERNANDES, Marcelo; CIRILO, Elder; AGNEZ, Lucas; BATISTA, Bruno; KULESZA, Uirá; ARANHA, Eduardo; and TREUDE, Christoph. Overcoming challenges in DevOps education through teaching methods. (2023). *Proceedings of the 45th International Conference on Software Engineering: Software Engineering in Practice, Melbourne, Australia, May 14-20*. 166-178.

Available at: [https://ink.library.smu.edu.sg/sis\\_research/8905](https://ink.library.smu.edu.sg/sis_research/8905)

This Conference Proceeding Article is brought to you for free and open access by the School of Computing and Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Computing and Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email [cherylds@smu.edu.sg](mailto:cherylds@smu.edu.sg).

---

**Author**

Samuel FERINO, Marcelo FERNANDES, Elder CIRILO, Lucas AGNEZ, Bruno BATISTA, Uirá KULESZA, Eduardo ARANHA, and Christoph TREUDE

# Overcoming Challenges in DevOps Education through Teaching Methods

Samuel Ferino<sup>1</sup>, Marcelo Fernandes<sup>1,2</sup>, Elder Cirilo<sup>3</sup>, Lucas Agnez<sup>1</sup>,  
Bruno Batista<sup>1</sup>, Uirá Kulesza<sup>1</sup>, Eduardo Aranha<sup>1</sup>, Christoph Treude<sup>4</sup>

<sup>1</sup>Federal University of Rio Grande do Norte, Brazil; <sup>2</sup>Federal Institute of Rio Grande do Norte, Brazil

<sup>3</sup>Federal University of São João del Rei, Brazil; <sup>4</sup>University of Melbourne, Australia

samuellucas97@ufrn.edu.br, marcelo.fernandes@ifrn.edu.br, elder@ufsj.edu.br, lucasagnez@hotmail.com  
brunokaike2014@gmail.com, {uira,eduardoaranha}@dimap.ufrn.br, christoph.treude@unimelb.edu.au

**Abstract**—DevOps is a set of practices that deals with coordination between development and operation teams and ensures rapid and reliable new software releases that are essential in industry. DevOps education assumes the vital task of preparing new professionals in these practices using appropriate teaching methods. However, there are insufficient studies investigating teaching methods in DevOps. We performed an analysis based on interviews to identify teaching methods and their relationship with DevOps educational challenges. Our findings show that *project-based learning* and *collaborative learning* are emerging as the most relevant teaching methods.

**Index Terms**—DevOps, teaching methods, challenges, mixed methods

## I. INTRODUCTION

DevOps (Development and Operations) appears as a natural evolution of the Agile movement [1], [2]. It aims to mitigate conflicts, approximate responsibilities, and improve the communication of Development and Operation teams [3]–[6]. This Agile perspective was essential for organizations to remain competitive in the technological business, meeting the need to build fast, resilient, and secure deliverables that involve distributed systems at scale. Recent studies show that the evolution of DevOps practices in industry organizations aligns with business success [7]. Since 2009, there have been various examples of large organizations adopting DevOps, including Facebook, Yahoo, Flickr, Netflix, and Etsy [8]–[10]. Using DevOps practices, companies such as Amazon and Netflix deploy thousands of times a day [11].

In many IT sectors, mastering DevOps practices has become a necessary skill [12], [13]. There is a high demand for DevOps practitioners, with many related job postings [14]. However, there is a shortage of qualified professionals to meet demand [15], [16]. Because of that, the industry has a growing desire to have graduates educated in the use of DevOps approaches and tools, better preparing students to apply DevOps practices in real projects [17], [18].

DevOps teaching has its challenges [19]. From an educational point of view, it is difficult to set up the DevOps approach from scratch [15], [18]. The challenge of creating an effective approach intensifies when considering students who do not have the necessary background to continuously evolve in learning. For example, students who have not worked in

industry or on projects of substantial size and complexity have difficulty understanding DevOps [20], [21]. More, DevOps research with an education focus is required to investigate these aspects.

Investing in DevOps teaching is a meaningful way to mitigate industry demand. From an educational perspective, the role of the educator is to guide students throughout the learning process by applying the proper teaching methods and providing a suitable learning environment [22].

Although there is no predominant approach, some approaches are more effective than others, depending on the teaching context [23]. *Project-based learning* teaching practice, for example, has been widely accepted as an important part of Software Engineering (SE) Education [24]–[26]. Educators and practitioners agree that the *lecture* approach (traditional teaching method) is not enough to teach SE [27]. Inactiveness of students, tiring lectures, one-way communication, and fast forgetting are the main disadvantages of this method [28], [29].

In this sense, approaching teaching from different perspectives comes from the idea that all people learn differently. Therefore, teaching must be considered as a changing variable. The way of approaching diversity in students also impacts the variety of teaching [27]. Another approach available to educators is to combine different teaching methods and approaches in a unique course. This idea is not new, as can be seen in the combination of theoretical (*lectures*) and practical (*lab*) classes. However, the challenge is to justify all teaching methods to fit into one course [30].

Ferino *et al.* [15] presented an initial study of teaching methods in DevOps Education. They analyzed 18 papers published up to 2019 from a systematic literature review of DevOps teaching experience. This study started the research on teaching methods in the DevOps area. However, more research is required to validate and improve the knowledge of DevOps teaching methods by educators, as this is a preliminary study.

This work aims to investigate the teaching methods that are being used in DevOps courses and how these methods address the challenges found in the DevOps Education area. To achieve this goal, we seek to answer the following research questions (RQs):

**RQ1.** *What are the teaching methods and approaches used in DevOps courses?*

**RQ2.** *How can teaching methods address the challenges of DevOps courses?*

These questions are answered by analyzing the data resulting from the study of Fernandes *et al.* [31], who interviewed 14 DevOps educators. That work analyzed challenges and recommendations in DevOps Education. Our paper focuses on identifying teaching methods and linking them to DevOps education challenges. As a result, this work provides the following contributions:

- Identification of 18 interview-derived teaching methods that can be implemented in DevOps courses to improve student learning. Our study identified 7 new teaching methods compared to a previous literature study [15]: *mentoring, review session, personalized learning, seminar, comprehensive distance learning, feedback session, and example-based learning.*
- A comprehensive set of 44 links between teaching methods and DevOps education challenges. Each link has a specific recommendation on how to use the teaching method to tackle the challenges. These links involve many teaching methods, different from the literature [15] that focuses on the links between challenges and the *educational support tool* teaching method.

The remainder of this paper is structured as follows. Section II presents related work. Then, Section III overviews the methodology used in this work. The report and discussion of teaching methods and related challenges occur in Sections IV and V. Section VI reviews the threats to the validity of this study. Final remarks about this work and discussions about research opportunities are presented in Section VII.

## II. RELATED WORK

Ferino *et al.* [15] present an initial study of teaching methods found in the literature, showing ways to help and mitigate challenges found in DevOps teaching. It relates to this work as it also looks for DevOps teaching methods, presenting 13 teaching methods. They investigated the *educational support tool* teaching method in more depth, focusing on linking it to educational DevOps challenges. Conversely, our study presents a total of 18 teaching methods identified analyzing DevOps educators' interviews. Additionally, we have validated the results, employing internal validation by member checking and external comparison with the study by Ferino *et al.* [15].

Grotta *et al.* [32] present a systematic literature review that investigates the performance of students with Information System (IS) teaching applied in DevOps. They show a correlation between teaching DevOps and professional and academic development. It also shows that DevOps is strongly related to *project-based learning*. That work is related to ours because it explores teaching methods applied to DevOps. However, that work focused on linking DevOps to the Information Systems area, while our work seeks to identify teaching methods and link them to the challenges found in DevOps courses.

Fernandes *et al.* [31] present an analysis of interviews with DevOps educators from around the world, collecting challenges and recommendations from existing DevOps courses according to their experiences. This work is related to ours, as it investigates challenges in DevOps teaching. Our work differs from it, as it focuses on identifying teaching methods and relating them to these challenges.

## III. METHODOLOGY

This section presents the methodology of this work, which is organized in two stages: (i) identifying the teaching methods used in DevOps courses; and (ii) linking these teaching methods to challenges. The basis for performing these analyzes is the data set produced by the study of Fernandes *et al.* [31], an interview study conducted with 14 DevOps educators. These educators have many years of experience teaching computing courses. During the interviews, they shared their experiences teaching DevOps courses. All these courses have DevOps as the main focus and address different DevOps aspects [33] such as *Runtime* and *Delivery*. Our work identifies the teaching methods based on their interview transcripts and links them to the challenges highlighted by Fernandes *et al.*

In every analysis for both stages, two people were involved, the main researcher and a reviewer researcher. Initially, the main researcher interprets the interview data and records observations and comments. The supporting researcher then reviews the records and discusses possible conflicting opinions with the main researcher. At this point, both researchers have to explain their reflections. A third researcher resolves the conflict if they do not achieve a common understanding. To help the researcher's work and to store the analyses and results, we used *Google Sheets*. We mention the interviewees using I terminology (I1, I2, ..., I14), where I10 refers to the tenth interview, for example.

### A. Identifying Teaching Methods

The first stage of this work is to identify teaching methods based on the 14 interview transcripts. We are adopting Westwood *et al.*'s [34] **teaching method definition**: *a set of principles, procedures, or strategies to be implemented by teachers to promote desired learning in students* [35], [36]. The following subsections provide more details on each step of this stage.

1) *Data Extraction*: The analysis starts by gathering mentions of teaching methods in the interviews' transcripts. For instance, consider the following interviewee's comment: "... I organize the students into teams of four or six per group, and then we work together. That's good also because they may be working in a team. They learn how to work in a team and team building..." (I11). The mention of "*They learn how to work in a team and team building*" is an example of the teaching method *collaborative learning* (definition in Table I). Furthermore, there are direct mentions of teaching methods, as in "*I use not only PBL, but also flipped classroom*" (I7). Interviewee I7 cites two teaching methods used in her DevOps course: *problem-based learning* (PBL) and *flipped classroom*.

2) *Data Analysis*: Based on the extracted data, we performed open coding through the identification and categorization of teaching methods. It is important to use codification, as teaching methods may not be explicitly mentioned. For example, the following text snippet refers to the *feedback session* teaching method: “... *I’m always taking almost an hour for students to give me their feedback ...*” (I12). We also observed whether the interviewees mentioned the use of teaching methods in their DevOps courses. Sometimes, we analyzed other parts of the interview transcripts to clarify the adoption of the methods. In this sense, sometimes it is not clear if the educator adopted *problem-based learning* during their course, as in the following quote: “*You always have to propose challenges to students*” (I4). When it is impossible to identify a text snippet that confirms the use of the teaching method during the course, the teaching method is considered not implemented.

3) *Feedback from Participants (member-check)*: We used a follow-up survey with the DevOps educators interviewed by Fernandes *et al.* [31] to evaluate the interpretations of the teaching methods used in the educators’ courses. Fernandes *et al.* had not provided any member checking. First, the survey instrument (online questionnaire) presents the aim of the work and defines what a teaching method is. The survey consisted of closed-ended and open questions. In closed-ended questions, educators indicate whether they agree or not with the teaching methods recognized in the interview transcripts. Answering these questions is mandatory. Open questions (optional) allow DevOps educators to communicate additional teaching methods that they often use during courses that were not recognized from the transcripts or to provide a link to course materials.

We sent the 14 personalized member-check surveys in May 2022 through their emails. After two weeks, we sent another email to reinforce the request for participants to contribute. We obtained feedback from 4 (28.5%) participants I2, I5, I9, and I13, getting 81.25% strong agreement for the identified teaching methods. I2 strongly agrees about using the following teaching methods: *collaborative learning*, *educational support tool*, *seminars*, and *labs*. He was neutral about *lectures* and *review sessions*. I5 strongly agrees with using *labs* and *project-based learning*. I9 strongly agrees with using case studies, *project-based learning*, and *collaborative learning*. He also explained that many case studies are related to how organizations have evolved their DevOps practices and the challenges they had to overcome. According to I9, these case studies do not focus on the technical aspects, but on the organizational and people aspects. I13 strongly agrees about using *project-based learning*, *lecture*, *tutorial*, and *collaborative learning*. He was neutral about using *labs*.

### B. Linking Teaching Methods to Challenges

We also linked the identified teaching methods to the challenges extracted from the interviews conducted by Fernandes *et al.* [31]. They presented 558 association links between recommendations and challenges as a means of solving or mitigating challenges. According to them, a challenge is defined as a problem that makes it difficult to plan or run a DevOps

course, while a recommendation is a real proposition to deal with a problem, easing the learning process. Some of their recommendations involve applying a teaching method such as “*Use of Problem-based Learning (PBL) for the teaching of DevOps.*” from I7. In this regard, we analyze the links between challenges and recommendations, checking whether the recommendation is related to a teaching method. In the case of any evidence, the researcher records the presence of a link between the teaching method and the challenge. Another researcher reviewer also checks the found links and discusses possible conflicting opinions, as explained at the beginning of the section.

We analyze the 558 links created by Fernandes *et al.* [31]. We identified 87 links involving the same respondent about the challenge and the recommendation. We also identified links that involved different respondents about the challenge and the recommendation. Our study is restricted to the analysis of links that involve the same interviewee in the challenge and recommendations. These links were more consistent, as they occurred in the interviewee’s courses.

## IV. RESULTS

In this section, we present the answers to the research questions of our study. We provide a set of artifacts [37] that contain all the data collected and analyzed in this paper.

### A. RQ1. WHAT ARE THE TEACHING METHODS AND APPROACHES IN DEVOPS COURSES?

Figure 1 shows the distribution of the 18 teaching methods identified in the interview transcript. We can observe that the four most cited methods are *project-based learning* (13 interviewee citations), *labs* (9 interviewee citations), *collaborative learning* (8 interviewee citations), and *lecture* (7 interviewee citations). On the other hand, the four least cited methods are *comprehensive distance learning*, *feedback sessions*, *flipped classroom*, and *tutorial* with only one interviewee citation each. We identified an average of 4 teaching methods by interview (maximum: 7; minimum: 2).

The rest of this subsection presents information about the identified teaching methods. We highlight how educators implemented those teaching methods in their respective courses.

**Project-based Learning.** This teaching method focuses on a project in which students work on a concrete task [38]. I5 motivates students to implement their projects from scratch. The students incrementally include DevOps practices (e.g., Automated Build, Automated Testing, and Continuous Integration) in the same project. However, the educator also considers giving a starter project to reduce the difficulties faced by students during the initial implementation of the project. The course project can also involve an open-source project in which students should have to improve by adding DevOps practices. In this context, I11 gives students the initial code to work on.

I1 proposes to develop a software project in incremental parts to teach students to deal with complexity. Different modules related to DevOps are required during incremental

TABLE I: Teaching methods definitions

Teaching Methods	Definitions
PROJECT-BASED LEARNING	It focuses on a project in which the students work on a concrete task [38].
COLLABORATIVE LEARNING	The students work collaboratively exchanging information and resolving tasks. The teacher is the active partner not just a repository of the information such as traditional education [39].
FLIPPED CLASSROOM	Activities traditionally conducted in the classroom become home activities (vice versa) [40].
SEMINAR	Students work in groups to discuss assigned questions and issues under the guidance of teachers. As an outcome of the seminars, the students give presentations or write an essay on their particular topic [30], [41].
LECTURE	It is the traditional teaching method in educational institutions in which the lecturer directly instructs the students [30], [42].
LABS	It involves accomplishing practical tasks exploring a computer science topic usually conducted in dedicated rooms equipped with computers for each student. [43].
REVIEW SESSION	In the review session, the answer (definition) is given and the student must come up with the question, a process that is the reverse of their study process [44].
PROBLEM-BASED LEARNING	It is a way of constructing and teaching courses using problems as the motivation and focus for students' activity [45].
PERSONALIZED LEARNING	It describes various instructional approaches aimed at meeting the learning needs of individuals [46]
EXPERIENTIAL LEARNING	It is the process of learning from concrete experience and reflecting about into real-world situations [47].
EDUCATIONAL SUPPORT TOOL	A tool or integrated environment used to support the teaching or learning [48].
COMPREHENSIVE DISTANCE LEARNING	It is an instructional model consciously selected in advance, with time to plan and make preparations to better ensure quality and accessibility of the learning experience for all students [49].
AGILE PROCESS	Use of Agile activities during course execution like sprints and scrum planning [50].
CASE STUDY	Practical examples encouraging students to integrate knowledge from class with real life [51].
FEEDBACK SESSION	The educator gives general feedback on the set of received assignments focusing on the aspects that most people did right, and the problems that were found [52].
MENTORING	It is an approach to improving teaching by adopting a mentor to help the educator during educational activities. A mentor helps the team organize their work and tracks if the team's planned didactic results are being achieved [53].
EXAMPLE-BASED LEARNING	It's based on providing worked examples that illustrate a written account of how a problem should be or can be solved [54].
TUTORIAL	A video or documentation with the purpose of to introduce the more general reader to the theoretical or technical concepts as well as configurations steps [55].

development, such as adding containerization with a unique container, then a cluster of containers.

I11 argues that students should work on innovative projects. I7 allows students to develop projects related to other areas of computer science, such as artificial intelligence, which are more attractive to them. I13 motivates students to contribute to open-source projects with more than 100 stars. I14 also encourages students to work on industrial projects that enable them to understand how to work in realistic scenarios and why mindset is essential.

I6 mentions the importance of planning the project requirements on which the students will work. This involves considering the programming language/frameworks and the associated tools. For example, they have adopted the Java language because the teaching staff has more confidence in this programming language. I6's course also adopts popular and recommended DevOps tools. Another requirement is to use tools that students can install and run on their computers. In this sense, I10 delimits a set of tools/technologies that

students could adopt in their projects, giving freedom only to the functional aspect of the projects. On the other hand, I13 gives students total freedom to choose the tools used in the project. I9 gives examples of projects to students to help them understand how to meet the project requirement. I11 summarizes the project requirements in a document.

I1 and I3 also argue the importance of a project-based approach as an alternative to course evaluation in the face of traditional exams. On the other hand, I4 has projects throughout the course, but does not evaluate them. I4 employs a project-based approach in which students are observed and the teacher tries to identify their difficulties and how they handle them. But I4 also uses the Net Promoter Score (NPS) to get feedback on the course from the students.

**Labs.** I3 explains that there are many ways to demonstrate a DevOps concept, supported by a large tools ecosystem. However, I5 reminds students that the course focuses on practicing the concepts with many tools, but not mastering these tools. In this context, I3 utilizes relevant free tools/technologies during

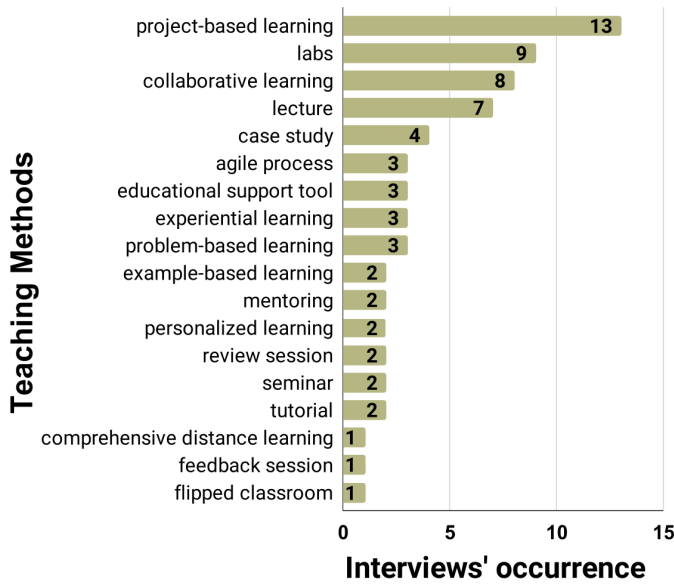


Fig. 1: 18 teaching methods distributed across 14 interviews.

labs. They use free services on public clouds. At the same time, I12 adopts open-source tools during labs. This enables students to practice in a realistic environment.

I8 automatizes the creation of the labs using snapshots. It facilitates the operation of the labs, mainly in large classes, where it is challenging to have personalized attendance. The educator also highlights the importance of observing the update of tools used in the labs, since new versions can break existing configurations. I10 and I14 faced the same problem. I11 and I12 report that the help of teaching assistants with proficiency in the tools has contributed to reducing the effort to prepare the labs.

**Collaborative learning.** Collaboration is one of the essential DevOps foundations. DevOps promotes collaboration between development and operation teams. I8 reinforces the importance of collaboration, as it is very common in the industry. I2 uses a Continuous Integration (CI) tool to track student collaboration in groups. I6 observes whether students work together or if each team member works alone. However, they think that it is not an efficient approach, as it takes a long time for the educator.

I6 uses problems and projects in groups to encourage student collaboration. I13, I11, and I8 use only team projects to teach how to work collaboratively in a team. I11 highlights that working in collaboration allows students to share problems with each other. I10 reinforces the need to be concerned about the differences between the environments of the team members. For example, differences in software versions could break the project build/pipeline.

I8 utilizes pair programming, where both students come up with the answers. I8 also creates a collaborative slack workspace. In this workspace, the educator and the students can send messages to each other. I8 explains to students that it is their responsibility as team members to keep their team

motivated.

**Lectures.** I2 shares recorded lectures with students, complemented by the virtual classroom. They use these virtual classrooms to answer and discuss student questions. I4 improves the dynamic of lectures by using strategies such as changing their voice and interacting with the students. I2 explains that it is easier for students not to focus in the classroom. I13 invites industrial guests to give part of the lectures to motivate and present students with real scenarios.

I11 utilizes lectures to teach and explain DevOps concepts and related challenges. They also introduce essential DevOps tools, such as Jenkins, a CI tool. I12 uses the lecture to share the experience and to detail specific DevOps concepts. They teach Agile and how to operate the Kanban board - DevOps concepts according to Leite *et al.* [33]. I12 also teaches containerization techniques through lectures.

**Other teaching methods.** I12 uses case studies from relevant companies such as Google, Meta, and Netflix to present a couple of industrial experiences in the classroom. I13 motivates the students to use the Katacoda<sup>1</sup> platform to share tutorials on the tools used throughout the course. I4 implements *personalized learning* through personalized support teams that support student challenges when setting up the tools on their machines.

**Combining teaching methods.** Table II shows the teaching methods distributed across the 13 courses. We categorized the teaching method occurrences in the interviews' transcripts into three levels: ● used in class, ◐ just as a recommendation, and ○ not identified. Each interview corresponds to a unique course, except I10 and I14 related to the 10th course. The black-gray color indicates that we found evidence of the teaching method adopted in the course. The light gray color indicates that the method was mentioned and that we could not find evidence of its implementation. In this context, we understand that the interviewee only recommends adopting the teaching method throughout the course. In this sense, 17 (94.4%) of the 18 identified teaching methods follow the implemented method category. As a highlight, *comprehensive distance learning* appears only in the category of recommended teaching methods. We also did not identify any use of teaching methods in the I6 interview, while I1 only implements *project-based learning*.

Analyzing Table II, it includes 13 combinations of implemented teaching methods together with teaching methods cited during interviews, without evidence of implementation. It also includes 12 combinations with only the teaching methods implemented during the class. For example, we identified *collaborative learning*, *mentoring*, *lecture*, *personalized learning*, *problem-based learning*, *project-based learning*, and *review session* during the interview with I4. We could identify evidence of implementation of *project-based learning*, *collaborative learning*, *review session*, *personalized learning*, and *mentoring* during their course.

<sup>1</sup><https://katacoda.com>

TABLE II: 18 teaching methods distributed across the 13 courses

Teaching Methods	Courses												
	1	2	3	4	5	6	7	8	9	10	11	12	13
PROJECT-BASED LEARNING	●	○	●	●	●	◐	●	●	●	●	●	○	●
COLLABORATIVE LEARNING	○	●	○	●	○	◐	○	●	◐	●	●	○	●
SEMINAR	○	◐	○	○	○	○	○	○	○	○	○	○	○
LECTURE	○	●	○	◐	○	○	○	○	○	●	●	●	●
LABS	○	●	●	○	●	○	○	●	○	●	●	●	●
REVIEW SESSION	○	●	○	●	○	○	○	○	○	○	○	○	○
PROBLEM-BASED LEARNING	○	○	○	◐	○	○	●	○	○	◐	○	○	○
PERSONALIZED LEARNING	○	○	○	●	○	○	○	●	○	○	○	○	○
EXPERIENTIAL LEARNING	○	○	○	○	○	○	◐	●	○	○	○	○	○
EDUCATIONAL SUPPORT TOOL	◐	●	○	○	○	○	●	○	○	○	○	○	○
COMPREHENSIVE DISTANCE LEARNING	○	○	○	○	○	○	◐	○	○	○	○	○	○
FLIPPED CLASSROOM	○	○	○	○	○	○	●	○	○	○	○	○	○
AGILE PROCESS	○	○	○	○	○	○	●	●	○	○	○	●	○
CASE STUDY	○	○	○	○	○	○	○	○	●	●	○	●	○
FEEDBACK SESSION	○	○	○	○	○	○	○	○	○	○	○	●	○
TUTORIAL	○	○	○	○	○	○	○	○	○	○	○	○	●
MENTORING	○	○	○	●	○	○	○	○	○	○	●	○	○
EXAMPLE-BASED LEARNING	○	○	●	○	●	○	○	○	○	○	○	○	○
<b>TOTAL</b>	2	6	3	8	3	2	7	6	3	7	5	5	5

Key: ● = used in class; ◐ = just as a recommendation; ○ = not identified.

**B. RQ2. HOW CAN TEACHING METHODS ADDRESS THE CHALLENGES IN DEVOPS COURSES?**

We identified 44 links between challenges and recommendations that involved the use of teaching methods. Figure 2 shows the most linked teaching methods that include *educational support tool*, *example-based learning*, *personalized learning*, *collaboration learning*, and *project-based learning*. We distinguish the DevOps-specific challenges using a ● background. Most challenges were linked to a unique teaching method, but we also have a few challenges linked to two teaching methods. For example, I3, I12, I13 and I14 shared the challenge that it is hard to teach DevOps concepts to students without industrial experience. Since DevOps motivation comes from solving conflicts between the development team and operations, students without industrial experience may not understand these conflicts. We identified *personalized learning* and *example-based learning* as teaching methods that address this challenge. I3 presents an example to students in a practical context from the initial stage to the final step. This worked example can complement students’ knowledge, providing a reference for them.

We could not identify links between challenges and recommendations for the following teaching methods: *flipped classrooms*, *seminar*, and *tutorials*. Next, we describe the links found between the challenges and the recommendations that represent teaching methods.

**Collaborative learning.** I8 and I10 share the challenge of teaching Agile techniques. They also comment on the difficulty in hands-on classes with 45 or more students. In this sense, I8 uses a team-based student organization that can help mitigate these challenges. The experience of working as a team enables students to reflect on the importance of collaboration, an Agile principle [56]. At the same time, it contributes to helping educators keep students focused since they are organized in groups.

I2 comments on the difficulty in teaching the student how to operate the system, allowing the addition of new features without breaking the existing system functionalities. They comment on the challenge of understanding the Continuous Delivery concept by using and practicing infrastructure environment setup. According to I2: “*The concept of Continuous Delivery [...] The difficult thing is to put it into practice [...] when they, as a team, need to release a certain functionality and ensure that it doesn’t break the system*”. For this reason, I2 teaches the students social coding, a collaborative learning approach through the socialization of knowledge [57]. In this context, social coding can motivate the students to learn interesting techniques and tools used by other student groups. This sharing of knowledge allows students to exchange information, produce ideas, simplify problems, and resolve tasks, as well as improve their understanding of DevOps practices.

**Personalized learning.** I3 and I7 share the perception that DevOps’ multidisciplinary nature is difficult to manage because it usually involves coverage of diverse software engineering disciplines. I3, I12, I13, and I14 share that teaching DevOps concepts to students without industrial experience is hard. According to I3: “*If the student is in a context where he has always been in the academic area, or he has never had practical experience with software development, [...] for the teacher, it becomes much more challenging to teach the DevOps concept to this student profile*”. For this purpose, I3 seeks to identify the most compatible DevOps scope for each class. They personalize the learning target according to the learning context of the students. Therefore, the educator should reflect and complement the initial level of knowledge of the students at the beginning of the course. For example, an educator can help students with little experience using Linux commands by sharing tutorials, creating labs, and encouraging discussions on DevOps topics.

I4, I5, I6, I8, and I10 commented that they had faced



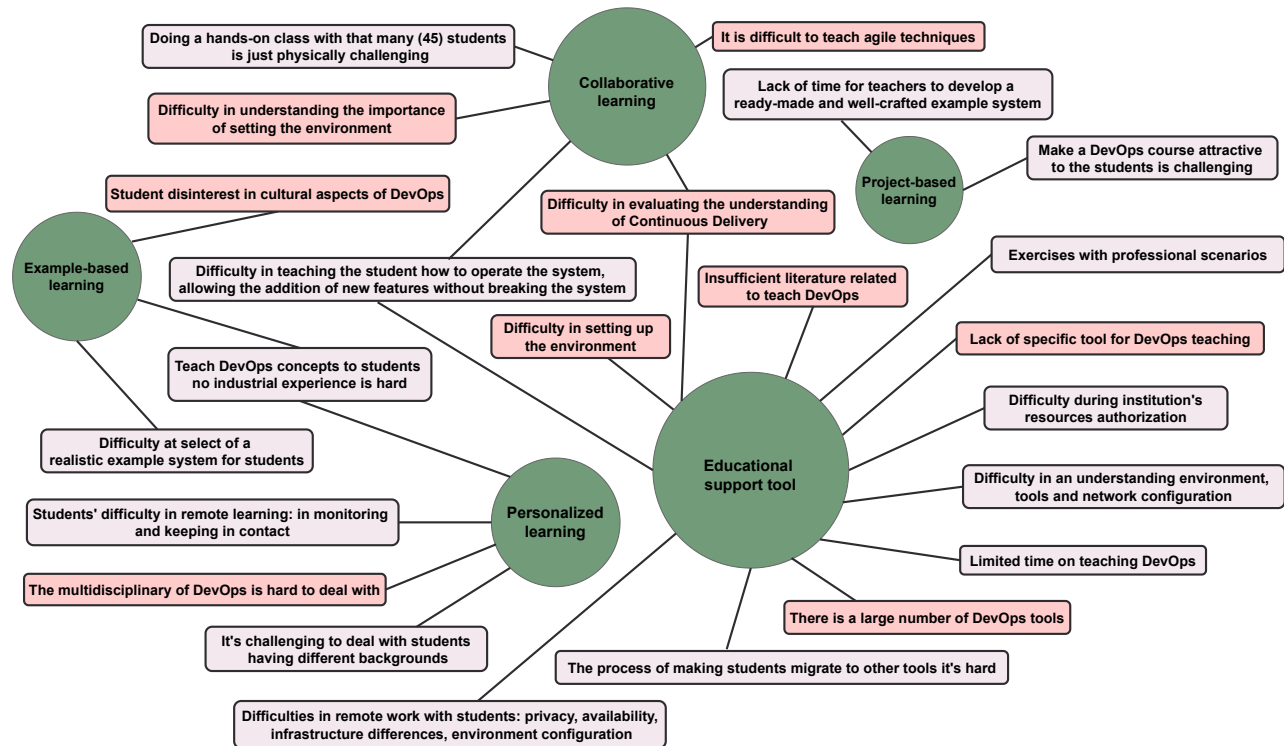


Fig. 2: Most linked teaching methods to DevOps challenges.

Key: ● = teaching method; ● = DevOps-specific challenge; ● = general challenge.

challenges in dealing with students of different backgrounds, making it difficult for students to collaborate in groups. According to I4: “One of the challenges is how do you teach people from these different backgrounds [...] there is so much technology that comes together in DevOps, that the challenge is how do you get everyone up to speed on an even level? So that we can all move forward together and learn together. So, that’s a big challenge”. I4 and I5 deal with it by seeking to know the student’s needs and limitations in advance. On the other hand, they suggest assuming that the students do not have much experience. I4 also avoids messing with student-specific problems, dealing with them in a personalized way at the right time. In other words, I4 alerts that the discussion of specific problems can eventually have a time limit; otherwise, the educator can lose the attention of the other students. I4 and I8 grade the students based on the student’s learning journey and mistakes. They argue that the essential is how the students get there, understanding that every failure is a learning opportunity.

To mitigate this challenge of different backgrounds, I4 and I8 seek to develop a communication culture between students and educators, allowing a more comfortable learning environment for students. Another challenge of using this personalized approach is the difficulty in interacting with students when classes are remote, which I4, I8, and I12 faced. This interaction involves monitoring student progress, cleaning and discussing doubts, and keeping in touch.

**Problem-based learning.** I7 describes the difficulty in

structuring the learning journey. I7 argues that the teaching plan should connect the covered DevOps subjects (for example, CI and Automated Tests). They use problem-based learning as a teaching method, starting from the problem and showing why people use it and what they are using. In this context, Meyer *et al.* [58] motivate the adoption of CI by facing the following main problem: “How do you verify whether someone’s changes broke something in the code or whether the changes work in the larger context of the entire codebase?”. Any educator could start teaching CI from this problem motivation and not only focus on teaching CI tools and their respective functionalities.

**Example-based learning.** I3, I4, I6, I9, I11, I12, and I14 emphasize that it is hard to show students that DevOps is not all about tooling (e.g., Docker and Jenkins). According to I3: “The student hopes to [...] learn that killer tool, which can help him in his professional career [...] he is usually more interested to know and learn the tools than understand the DevOps culture”. There is little interest in learning the DevOps cultural aspects, such as collaboration. For this purpose, I3, I4, I11, and I12 use concrete examples during the teaching. These examples do not focus on a specific tool. Students should understand that there is no *silver bullet* and that each problem can have a set of comprehensive solutions.

Another challenge mitigated by the *example-based learning* approach is that teaching DevOps concepts to students without industrial experience is difficult. Concrete examples can contribute to clarifying the student’s understanding. However,

I5, I9, I11, and I14 highlight the difficulty in selecting realistic examples of systems for students. According to I5: “*The challenge sometimes is finding a good open-source application, which is not too big also because you don’t want the project to be too big. You don’t want it to be too small, but you don’t want too big*”. Simple systems can demotivate students. For this purpose, I5 plans to provide a ready-made sample system to students. As a benefit, the educator can have more confidence in supporting students during the course.

I11 and I12 share the challenge of creating an attractive DevOps course from the students’ perspectives. I11 worries that the class project should not be very small and must be challenging. Small projects such as toy projects may not be attractive to students since they are usually simple.

**Discussion.** I3, I4, I6, and I8 comment on the challenge of teaching DevOps culture. They also comment that there is no ready-made recipe (step-by-step instructions) to teach the DevOps mindset. DevOps culture includes aspects such as continuous feedback and sharing of knowledge [59]. I3 and I4 dealt with this challenge by promoting discussions of DevOps concepts and related issues. They use the students’ difficulties, opinions, and experiences, pointing out solutions using DevOps. By doing this, the students could familiarize themselves more easily since the explanation is related to something they know well.

**Mentoring.** I4 shares difficulties in remote teaching. He highlights that students need to worry about setting up their computers while studying at home. It includes installing the necessary software. In this context, I4 created a specific support team to deal with infrastructure queries. This team helps students solve software infrastructure problems.

I7 mentions the need for great effort during the assessment of large classes. For this purpose, I7 highlights the importance of teaching assistants to help in the assessment process: “*team of teaching assistants [...] If you don’t have them, it’s more difficult, you need to assess the projects alone. Take a class with 40 students, even if you divide it into teams, it’s a lot of work...*”. Teaching assistants can help the leading educator.

**Agile processes.** I8 and I10 highlight the difficulty in teaching Agile techniques, as already mentioned. However, another approach to mitigate this challenge is to make the course Agile-oriented. I8 organizes the course in sprints with challenging incremental deliverables. In this context, we suggest the adoption of Agile techniques such as planning meetings and daily discussions [56] to make the student understand Agile processes and techniques through experimentation.

**Case study.** As mentioned above, I3, I4, I6, and I8 comment on the challenge of teaching DevOps culture. In this sense, I3 presents DevOps case studies in their course to students to help them understand cultural aspects. These case studies include the elimination of silos between development and operations teams; and the relationship between DevOps and the Site Reliability Engineering (SRE) professional.

**Project-based learning.** I11 reports that making a DevOps course attractive to students is a challenge: “*You can make the lectures more interactive, but to make the lecture attractive*

*students have to be willing to interact. [...] Which is very difficult to do*”. For this purpose, they argue that the project of the class should not be very small. Amorim *et al.* [60] identified a lack of opportunity for student creativity, using small projects.

**Feedback Session.** We already commented that I8 and I10 had difficulty executing the hands-on classes with 45 or more students. In I8’s course, they have teaching assistants (TA) who help students with the management of technical questions related to the exercises. After the class, the TAs answer the remaining questions that are not handled during the class. They used an online shared document where all students could improve their understanding by reading the TAs’ feedback.

**Labs.** I1, I3, I5, I6, I8, I9, I10, and I11 mentioned the challenge of not having enough time in the course to teach DevOps. For example, I3 mentions: “*the issue of laboratories [...] you always end up as a matter of time versus class development*”. This challenge occurs because DevOps incorporates knowledge of many practices. To illustrate, Leite *et al.* [33] developed a DevOps conceptual map that includes more than one hundred concepts. In this context, I1, I8, I10, and I11 seek to focus more classes on the practical part than DevOps’s theoretical part (e.g., DevOps mindset). They recommend that the courses have a practical part that occupies at least 80% of the classes. Then, the understanding of the students can be improved through experimentation.

**Lecture.** I12 alerts about the gap between what the industry wants from students about DevOps and what the university teaches: “*We hear from our industrial partners and from industry in general there’s this huge gap between what the industry needs and what university provides*”. I12 seeks to mitigate this by carefully selecting industrial speakers to share their experiences with students. According to I12, inviting these experts allows the student to face real problems and scenarios. This enables students to reflect on the importance of DevOps practice in the industry.

**Educational support tool.** I5 comments that many students have difficulty understanding the practice of operations (for example, network configuration). These students have considerable proficiency in software development practices, but low proficiency in Linux commands and operations practices. They may also face problems configuring their machines and installing infrastructure software since they are not used to practicing it. In this context, I5 provides an initial configuration of the environment for students. They use a virtual machine snapshot that contains the required course software. It includes a containerization tool (e.g., Docker), CI/CD server (e.g., Jenkins), and artifact repository (e.g., Artifactory).

I2, I5, and I11 mentioned that the preparation of the course is challenging, since there is insufficient literature on teaching DevOps. For example, there is insufficient material to help DevOps educators prepare their courses. I2 also highlights the difficulties in assessing the student’s understanding of continuous delivery techniques. To mitigate this challenge, they use an integrated environment in which students put their solutions to the exercises.

On the other hand, I2 exposes the challenge of making clear to students the importance of having a more realistic perspective of production. To address it, they adopt industrial tools in the curriculum. Students can learn DevOps concepts by practicing with the same tools used in companies such as Google and Microsoft, facing and reflecting on real problems. This approach facilitates the preparation of students for the industry.

However, I2 commented that they faced difficulties in requiring students to adopt course tools. Many students are resistant to adopting these tools. I2 explains that students usually do not feel comfortable using new tools. To mitigate it, the educator explains to the students that they have a good grasp of the features of the tool and that the students will have good support for any questions. On the other hand, many educators (I2, I3, I5, I6, I7, I8, and I11) mention a related challenge: the educator (and teaching assistants) should be able to manage tools for each step of the DevOps pipeline taught in their courses.

Finally, I1 mentioned the difficulty of holding an institution’s resources authorization. According to I1: “*challenges are basically how do you rebuild an enterprise environment into a university environment that is much more restrictive and doesn’t have enough machines for them*”. This restriction includes, for example, the inability to install tools and network limitations to create near-real infrastructure as related challenges. For this reason, I1 uses cloud providers’ services as infrastructure during the course. Although these cloud services usually require payment, I1 recommends that students use the students’ free plan. I1 defends this approach by: “*I recommend [...] Moving all teaching to a cloud. [...] contact AWS. They have a student program, or Google, with Ali Baba, Azure, and IBM Cloud*”.

## V. DISCUSSION

We discuss our results by comparing them with the findings presented by Ferino *et al.* [15] and in terms of the implications of the results for educators and researchers.

### A. Comparing Teaching Methods From Interviews and Literature

We compared our results with a systematic literature review that investigates existing DevOps teaching methods conducted by Ferino *et al.* [15]. Their study selects a total of 18 papers reporting DevOps teaching experiences until the beginning of 2020. In this context, we first update Ferino *et al.*’s results [15] by following their SLR methodology, including more 15 papers published between 2020 and 2021. We finished with a total of 33 papers, referenced in our online appendix [37].

Figure 3 presents the 21 teaching methods identified in the SLR considering our update. The four most cited methods are *project-based learning* (14 paper citations), *lecture* (13 paper citations), *collaborative learning* (12 paper citations), and *Agile process* (10 paper citations). On the other hand, the three least cited methods are *studio-based learning*, *game-based learning*, and *research-based teaching* with only one

paper citation each. They refer to unusual teaching methods. *Game-based learning* refers to the use of computer games and games in general for educational purposes [39]. *Studio-based learning* focuses on student activity that involves interactions with peers and instructors. Studios are experiential learning opportunities based on projects / products. In studios, students discover for themselves what content is needed, what questions need to be answered, and what the answers are [61]. *Research-based teaching* refers to students actively exploring and solving problems using scientific research approaches under the guidance of educators [62]. Finally, we identified an average of 3 teaching methods by paper (maximum: 8; minimum: 1).

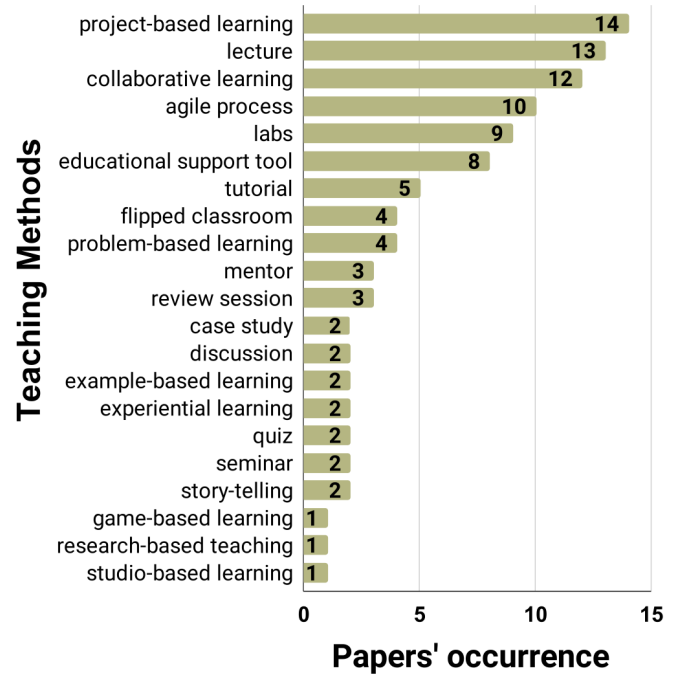


Fig. 3: Distribution of 21 teaching methods by 33 papers.

Our comparison identified a set of similar teaching methods from both sources: interviews and papers. As a highlight, both emphasize that *project-based learning* and *collaborative learning* are the most recurrent teaching methods. Most of the current DevOps courses focus on letting students work together on projects. This is in line with industry needs: professionals who have both hard and soft skills [63]. Hard skills are mastered through working on projects, while soft skills proficiency is developed through collaborative work. Rabelo *et al.* [63] identified *teamwork* as the most in-demand non-technical skill in software development jobs today. Shih *et al.* [64] found that IT professionals often have introverted personalities. They show that this can be a barrier to understanding the needs and requirements of the customer [63]. Collaborative work helps students improve their communication skills. For this purpose, DevOps courses today have a great impact on the formation of students as IT professionals.

We identified only four studies [65]–[68] investigating how effective these teaching methods are in the context of DevOps

Education. Many studies only have student feedback surveys as a teaching method evaluation approach [69]–[78]. The absence of evaluation studies on the adoption of teaching methods in DevOps education represents an opportunity for researchers.

Eddy *et al.* [65] present an empirical study that evaluates whether the use of a Continuous Delivery (CD) pipeline in a laboratory environment helps students better understand CD-related concepts. They found that the pipeline and activity were useful for students’ understanding of the concepts. Okolica *et al.* [68] conducted a pilot study on *game-based learning* in a graduate-level course, where the lesson objectives focus on DevSecOps. The game was tested at the end of the course. Their results show that the students consider the game relevant to illustrate the topic.

Perez *et al.* [66] analyze whether the use of the *research-based teaching* (RBT) experience was effective for student learning. They also analyze whether inter-coder agreement (ICA) can be applied to verify student knowledge acquisition. As a result, the students showed homogeneous academic performance compared to the traditional-learning student group. However, they argue that the DevOps RBT experience has an inherent advantage in providing motivational learning as students acquire knowledge from their research. Abirami *et al.* [67] present an experimental study of applying various teaching methods in a DevOps course. They focused on mixing the following learning approaches: *discussions*, *seminars*, and *quizzes*. Unfortunately, we only identified this study working on blended learning in DevOps Education. They used the *logistic regression model* to measure the impact of the blended learning approach on student learning. They identified that when these teaching methods are used, students have a significant impact on increasing their learning outcomes. This approach emerges as a starting point for new research that measures the impact of integrated teaching methods on student learning. Next, we describe the similarities and differences in applying the teaching methods reported in the interviews and literature.

**Similarities.** We confirm that 15 (83.3%) of the 18 teaching methods derived from the interviews were also identified in the literature review. *Lectures* are highlighted in both sources. This is a simple, fast, and inexpensive method to present the vast issues to many groups of learners [28], [79]. In remote teaching, *lectures* is highlighted as a low-cost approach. In this context, Hobeck *et al.* [80] mentioned that *flipped classroom* facilitated the transition to fully remote teaching during the COVID pandemic. They reported that the pre-class activities were unchanged since the lectures were already recorded on video, although the in-class activities changed significantly.

**Differences.** Our analysis identified that the following teaching methods were explored only in the papers of the SLR but not in the interviews of existing DevOps courses: *storytelling*, *game-based learning*, *research-based teaching*, *studio-based learning* and *quizzes*. We understand that educators could not be familiar or secure enough about how to implement these approaches, since DevOps is a new area. On

the other hand, *personalized learning*, *comprehensive distance learning*, and *feedback session* appear only in the interviews.

Christensen *et al.* [69], [72] use *storytelling* to design the learning context of DevOps topics in an existing undergraduate course. In this context, Rao *et al.* [81] emphasize the potential of *storytelling* as a tool to improve *lectures*, engaging the student through stories. Ouhbi *et al.* [82] conducted an empirical study exploring the effects of combining *storytelling* with *lectures*. They identified a great impact on female students, increasing their interest in pursuing a career in software engineering. Therefore, this approach can be adopted as a strategy to engage female students, increasing female participation in the area of software engineering.

We also identified interesting recommendations on how to apply teaching methods. For example, Krusche *et al.* [71] implement *project-based learning* adopting projects with real customers in the industry. The students implemented Continuous Delivery pipelines on customer projects, enabling fast customer feedback. Wei *et al.* [83] and Alves *et al.* [74] motivate the students to work on open-source projects. This enables students to collaborate using DevOps tools. They represent projects with potential customers without a roadmap, similar to real software projects. Alves *et al.* argue that project restrictions are important in limiting the scope of the project. In this course, the students present a set of project themes, but the students can also work on other project themes, as long as they are approved. However, Wei *et al.* let the students be free to use whatever technology they choose.

## B. Lessons Learned for Educators

Next, we discuss the lessons learned, focused on improving the teaching experience.

**Teaching DevOps using Practical Approaches.** Similar to Grotta *et al.* [32], our findings show that DevOps concepts should be taught using practical teaching methods. Teaching methods such as *project-based* and *problem-based learning* appear to be good choices. Using these approaches, a DevOps course can contribute to improving the problem-solving skills of students. Moreover, *project-based learning* and *collaborative learning* emerged as a recurrent teaching method combination. We notice the potential of this approach to be implemented as a basis of the DevOps course.

**Collaborative Learning fits with the Culture of Collaboration Principle.** In the paper, we have mentioned the importance of promoting collaboration practices in DevOps courses. Adopting *collaborative learning* allow students to understand the challenges of working in a team (e.g., keeping group members motivated) and overcome collaboration difficulties with the support of educators.

**Establishing Industry-Academia Collaboration.** Industrial *mentoring* and industrial guest lectures appear as a potential opportunity for starting collaboration between industry and academia. Although there are challenges involving external collaboration, industrial practitioners are usually more proficient in technologies than academics. This approach enables the students to learn the technical skills needed for the industry

in a realistic scenario. The industry can also supply the demand for professionals capable of facing their challenges.

### C. Implications to Researchers

Next, we discuss research opportunities for new studies.

**Recommending New Teaching Methods.** Trede *et al.* [61] suggest that more innovative learning and teaching approaches in engineering education can help prepare students for the future world of work. Our study identified a set of teaching methods. However, we urge researchers to seek to increase this set by adopting and combining different teaching methods. We recommend starting research with evaluated teaching methods in software engineering and other areas of computer science. We also recommend focusing on more active and practice-oriented approaches, as DevOps seems to fit these strategies.

**How DevOps Courses Impact Students' Non-Technical Skills.** Our study identified an expressive adoption of collaborative learning in DevOps courses. Trede *et al.* [61] identified that this teaching method is closely related to teamwork and leadership. These non-technical skills are very important in the industry. Thus, we reinforce the importance of teaching collaboration concepts in the DevOps courses as part of the computer science and/or software engineering academic curricula.

**More Adequate Teaching Approaches for Training in Industry.** We identified two papers [73], [84] that focus on teaching DevOps in an industrial context. Both studies did not conduct empirical studies on the integration of teaching methods. Teaching in an industrial context has hard time constraints, with training duration no longer than a few weeks. For this purpose, an interesting research topic is to understand what combination of teaching methods will more effectively contribute to teaching DevOps concepts to employees of a software company.

## VI. THREATS TO VALIDITY

In this section, we discuss threats to the validity of this study in the context of qualitative research [31], [85]–[87].

**Credibility.** This validity refers to whether the study findings are correctly drawn from the original data. We employed several approaches to ensure credibility: (a) we always had a second researcher reviewing the data analyzed by the first researcher during the execution of the study; (b) we carried out member checking on the teaching methods with the participants, allowing them to complement our analysis with teaching methods not previously identified; (c) we compared our findings from interviews with Ferino *et al.* [15]'s findings from SLR, including more studies by following SLR's methodology.

**Transferability.** It is related to the extent to which our results can be transferred to other contexts (generalization). Our study extends Fernandes *et al.* [31]'s transferability limitation since we analyzed their interviews' transcripts. They selected 14 DevOps educators from three continents (North America, South America, and Europe) focusing on diversity, covering academia and industry.

**Confirmability.** It refers to the degree to which other researchers can verify the findings. We show the evidence for each identified teaching method by quoting participants. We share our data sheets including the teaching methods and links between the teaching methods and the challenges in our online research artifact [37].

## VII. CONCLUSION

Our study aims at recognizing teaching methods used in the planning and execution of DevOps courses. We identified 18 teaching methods from interviews conducted by Fernandes *et al.* [31]. In summary, our study revealed that educators should favor practical methods to teach DevOps. Indeed, *project-based learning* was the most recurrent teaching method reported in the interviews. Also, as observed in the literature review, most DevOps courses focus on qualifying students to work together. Likewise, other teaching methods (e.g., *problem-based learning*) appear to be sound alternatives for improving students' problem-solving skills.

Our work also introduced 44 links between teaching methods and DevOps education challenges. *Educational support tool*, *collaborative learning*, *personalized learning*, *example-based learning*, and *project-based learning* are the most used teaching methods by educators in their classes to mitigate the challenges. As reported in this work, there is a recurrent challenge in teaching DevOps culture. The promotion of discussion enables educators to use students' experiences to explain how DevOps could handle the issues faced. The students also face difficulty in implementing Continuous Delivery practices, which is an essential DevOps concept. A collaborative approach using social coding appears as a mitigation strategy. The students can improve their skills by learning interesting techniques from other students.

Finally, we also examined research opportunities — e.g., the recommendation of new teaching methods and learning considering non-technical skills in DevOps courses. We aim to explore the studied teaching methods and approaches for training DevOps professionals in the industry as future work.

**Acknowledgements.** We thank all the educators who contributed to our study. This work is partially supported by INES (www.ines.org.br), CNPq grant 465614/2014-0, CAPES grant 88887.136410/2017-00, FACEPE grants APQ-0399-1.03/17, PRONEX APQ/ 0388-1.03/14, and IFRN.

## REFERENCES

- [1] F. Almeida *et al.*, "Exploring the benefits of combining devops and agile," *Future Internet*, vol. 14, no. 2, p. 63, 2022.
- [2] E. Bobrov, A. Bucchiarone, A. Capozucca, N. Guelfi, M. Mazzara, A. Naumchev, and L. Safina, "Devops and its philosophy: Education matters!" in *Microservices*. Springer, 2020, pp. 349–361.
- [3] I. S. Committee *et al.*, "Ieee standard for devops:building reliable and secure systems including application build, package, and deployment," *IEEE Std 2675-2021*, pp. 1–91, 2021.
- [4] K. Bahadori *et al.*, "Devops meets dynamic orchestration," in *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer, 2018, pp. 142–154.

- [5] P. Rodrigues *et al.*, “Devops adoption in junior enterprise: an experience report of software development,” in *Anais da 1 Escola Regional de Engenharia de Software*. SBC, 2017, pp. 89–96.
- [6] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, “Devops,” *Ieee Software*, vol. 33, no. 3, pp. 94–100, 2016.
- [7] N. Forsgren, J. Humble, and G. Kim, “Accelerate: The science of lean software and devops building and scaling high performing technology organizations,” 2018.
- [8] G. Rong, S. Gu, H. Zhang, and D. Shao, “Devopsenvy: an education support system for devops,” in *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 2017, pp. 37–46.
- [9] F. M. Erich *et al.*, “A qualitative study of devops usage in practice,” *Journal of Software: Evolution and Process*, vol. 29, no. 6, p. e1885, 2017.
- [10] S. K. Bang, S. Chung, Y. Choh, and M. Dupuis, “A grounded theory analysis of modern web applications: knowledge, skills, and abilities for devops,” in *Proceedings of the 2nd annual conference on Research in information technology*, 2013, pp. 61–62.
- [11] A. Agarwal *et al.*, “Continuous and integrated software development using devops,” in *2018 International conference on advances in computing and communication engineering (ICACCE)*. IEEE, 2018, pp. 290–293.
- [12] M. Wiesche, “Are you ready for devops? required skill set for devops teams devops teams,” in *Twenty-Sixth European Conference on Information Systems (ECIS2018)*, 2018, pp. 12–14.
- [13] D. Spinellis, “Being a devops developer,” *IEEE Software*, vol. 33, no. 3, pp. 4–5, 2016.
- [14] C. Pang, A. Hindle, and D. Barbosa, “Understanding devops education with grounded theory,” in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training*, 2020, pp. 107–118.
- [15] S. Ferino, M. Fernandes, A. Fernandes, U. Kulesza, E. Aranha, and C. Treude, “Analyzing devops teaching strategies: An initial study,” in *Brazilian Symposium on Software Engineering*, 2021, pp. 180–185.
- [16] S. Krusche *et al.*, “Introduction of continuous delivery in multi-customer project courses,” in *Companion Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 335–343.
- [17] R. A. K. Jennings *et al.*, “Devops-preparing students for professional practice,” in *2019 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2019, pp. 1–5.
- [18] E. Bobrov *et al.*, “Teaching devops in academia and industry: reflections and vision,” in *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer, 2019, pp. 1–14.
- [19] H. B. Christensen, “Teaching devops and cloud computing using a cognitive apprenticeship and story-telling approach,” in *Proceedings of the 2016 ACM conference on innovation and technology in computer science education*, 2016, pp. 174–179.
- [20] C. Jones, “A proposal for integrating devops into software engineering curricula,” in *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer, 2018, pp. 33–47.
- [21] J. Perez, A. Gonzalez-Prieto, J. Diaz, D. Lopez-Fernandez, J. Garcia-Martin, and A. Yague, “Devops research-based teaching using qualitative research and inter-coder agreement,” *IEEE Transactions on Software Engineering*, 2021.
- [22] T. Kilamo *et al.*, “Teaching collaborative software development: A case study,” in *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 2012, pp. 1165–1174.
- [23] M. R. Marques *et al.*, “A systematic mapping study on practical approaches to teaching software engineering,” in *2014 IEEE Frontiers in education conference (FIE) proceedings*. IEEE, 2014, pp. 1–8.
- [24] X. Bai *et al.*, “The devops lab platform for managing diversified projects in educating agile software engineering,” in *Proceedings of the FIE 2018*. IEEE, 2018, pp. 1–5.
- [25] R. Chatley *et al.*, “Lean learning-applying lean techniques to improve software engineering education,” in *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*. IEEE, 2017, pp. 117–126.
- [26] A. M. Moreno *et al.*, “Enriching traditional software engineering curricula with software project management knowledge,” in *Proceedings of the 38th International Conference on Software Engineering Companion*, 2016, pp. 404–411.
- [27] J. Chen, H. Lu, L. An, and Y. Zhou, “Exploring teaching methods in software engineering education,” in *2009 4th International Conference on Computer Science & Education*. IEEE, 2009, pp. 1733–1738.
- [28] R. Sadeghi *et al.*, “Comparison of the effect of lecture and blended teaching methods on students’ learning and satisfaction,” *Journal of advances in medical education & professionalism*, vol. 2, no. 4, p. 146, 2014.
- [29] H. Norouzi and S. Mohsenizadeh, “Jafary suny h, ebrahimzadeh s. the effect of teaching using a blend of collaborative and mastery of learning models, on learning of vital signs: an experiment on nursing and operation room students of mashhad university of medical sciences,” *Iranian Journal of Medical Education*, vol. 11, no. 5, pp. 544–53, 2011.
- [30] M. Kuhmann, “A practical approach to align research with master’s level courses,” in *2012 IEEE 15th International Conference on Computational Science and Engineering*. IEEE, 2012, pp. 202–208.
- [31] M. Fernandes, S. Ferino, A. Fernandes, U. Kulesza, E. Aranha, and C. Treude, “Devops education: An interview study of challenges and recommendations,” in *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 2022, pp. 90–101.
- [32] A. Grotta and E. P. V. Prado, “Devops didactic transposition in is higher education: A systematic literature review,” in *Proceedings of the Americas Conference on Information Systems: SIG ED - IS in Education, IS Curriculum, Education and Teaching Cases*, 2022.
- [33] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, “A survey of devops concepts and challenges,” *ACM Computing Surveys (CSUR)*, vol. 52, no. 6, pp. 1–35, 2019.
- [34] P. S. Westwood, *What teachers need to know about teaching methods*. Aust Council for Ed Research, 2008.
- [35] G. Björk Stiernström, “Video tutorials for artists in the entertainment industry: A comparative study of what makes a video tutorial effective,” 2017.
- [36] Q.-x. Liu and J.-f. Shi, “An analysis of language teaching approaches and methods—effectiveness and weakness,” *Online Submission*, vol. 4, no. 1, pp. 69–71, 2007.
- [37] S. Ferino, M. Fernandes, E. Cirilo, L. Agnez, B. Batista, U. Kulesza, E. Aranha, and C. Treude. (2022) Overcoming challenges in devops education through teaching methods. [Online]. Available: <https://doi.org/10.5281/zenodo.7135611>
- [38] M. Indiramma, “Project based learning—theoretical foundation of computation course,” in *2014 International Conference on Interactive Collaborative Learning (ICL)*. IEEE, 2014, pp. 841–844.
- [39] M. Pivec, O. Dziabenko, and I. Schinnerl, “Aspects of game-based learning,” in *3rd International Conference on Knowledge Management, Graz, Austria*, vol. 304, 2003.
- [40] G. Akçayır and M. Akçayır, “The flipped classroom: A review of its advantages and challenges,” *Computers & Education*, vol. 126, pp. 334–345, 2018.
- [41] H. L. Zeng, D. X. Chen, Q. Li, and X. Y. Wang, “Effects of seminar teaching method versus lecture-based learning in medical education: a meta-analysis of randomized controlled trials,” *Medical Teacher*, vol. 42, no. 12, pp. 1343–1349, 2020.
- [42] B. O. Omatseye, “The discussion teaching method: An interactive strategy in tertiary learning,” *Education*, vol. 128, no. 1, 2007.
- [43] O. Hazzan, N. Ragonis, and T. Lapidot, “Lab-based teaching,” in *Guide to Teaching Computer Science*. Springer, 2020, pp. 221–249.
- [44] V. M. Harder and G. L. Burns, “Geopardy—a computer-assisted review session,” *Journal of Geological Education*, vol. 40, no. 3, pp. 222–226, 1992.
- [45] I. Richardson and Y. Delaney, “Problem based learning in the software engineering classroom,” in *2009 22nd Conference on Software Engineering Education and Training*. IEEE, 2009, pp. 174–181.
- [46] Y. Akyuz, “Effects of intelligent tutoring systems (its) on personalized learning (pl),” *Creative Education*, vol. 11, no. 6, pp. 953–978, 2020.
- [47] D. Kolb, *Experiential learning: experience as the source of learning and development*. Englewood Cliffs, NJ: Prentice Hall, 1984.
- [48] F. Shahid, M. Aleem, M. A. Islam, M. A. Iqbal, M. M. Yousaf *et al.*, “A review of technological tools in teaching and learning computer science,” *Eurasia Journal of Mathematics, Science and Technology Education*, vol. 15, no. 11, p. em1773, 2019.
- [49] A. J. Maletz, “Teaching in the dark: Increasing student engagement in comprehensive distance learning,” Master’s thesis, Western Oregon University, Monmouth, 2021.

- [50] P. Salza, P. Musmarra, and F. Ferrucci, *Agile Methodologies in Education: A Review*. Singapore: Springer Singapore, 2019, pp. 25–45.
- [51] S. Krusche, N. von Frankenberg, and S. Afifi, “Experiences of a software engineering course based on interactive learning,” in *SEUH*, 2017, pp. 32–40.
- [52] V. Isomöttönen, V. Tirronen, and M. Cochez, “Issues with a course that emphasizes self-direction,” in *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, 2013, pp. 111–116.
- [53] S. DOWDALL, A. HĽOBAŽ, P. MILCZARSKI, D. O’REILLY, K. PODLASKI, and Z. STAWSKA, “Multinational, intercultural, multidisciplinary and intensive (mimi) methodology to enrich soft skills development in computer science students,” *Informatics in Education*, vol. 20, no. 3, pp. 391–420, 2021.
- [54] T. Van Gog and N. Rummel, “Example-based learning: Integrating cognitive and social-cognitive research perspectives,” *Educational psychology review*, vol. 22, no. 2, pp. 155–174, 2010.
- [55] E. Lokkila, E. Kaila, V. Karavirta, T. Salakoski, and M. Laakso, “Redesigning introductory computer science courses to use tutorial-based learning,” *EDULEARN16 Proceedings*, pp. 8415–8420, 2016.
- [56] P. Diebold and M. Dahlem, “Agile practices in practice: a mapping study,” in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, 2014, pp. 1–10.
- [57] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, “Social coding in github: transparency and collaboration in an open software repository,” in *Proceedings of the ACM 2012 conference on computer supported cooperative work*, 2012, pp. 1277–1286.
- [58] M. Meyer, “Continuous integration and its tools,” *IEEE software*, vol. 31, no. 3, pp. 14–16, 2014.
- [59] M. Sánchez-Gordón and R. Colomo-Palacios, “Characterizing devops culture: a systematic literature review,” in *International Conference on Software Process Improvement and Capability Determination*, 2018, pp. 3–15.
- [60] L. F. Amorim, “How (un) happiness impacts on software engineers in agile teams?” *International Journal of Software Engineering & Applications (IJSEA)*, vol. 11, no. 3, 2020.
- [61] F. Trede, R. Braun, and W. Brookes, “Engineering students’ expectations and perceptions of studio-based learning,” *European Journal of Engineering Education*, vol. 46, no. 3, pp. 402–415, 2021.
- [62] L. Ye, W. Liu, and J. Jiang, “Exploration and practice of research-based teaching on basic chemistry experiment in local colleges and universities,” in *2017 7th International Conference on Education, Management, Computer and Society (EMCS 2017)*. Atlantis Press, 2017, pp. 982–986.
- [63] D. Rabelo, A. Lopes, W. Mendes, C. de Souza, K. Gama, D. Monteiro, and G. Pinto, “The role of non-technical skills in the software development market,” in *Proceedings of the XXXVI Brazilian Symposium on Software Engineering*, 2022, pp. 31–40.
- [64] S.-P. Shih, T. Lie, G. Klein, and J. J. Jiang, “Information technology customer aggression: The importance of an organizational climate of support,” *Information & management*, vol. 51, no. 6, pp. 670–678, 2014.
- [65] B. Eddy, N. Wilde, N. Cooper, B. Mishra, V. Gamboa, K. Shah, A. Deleon, and N. Shields, “A pilot study on introducing continuous integration and delivery into undergraduate software engineering courses,” in *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 2017, pp. 47–56.
- [66] J. Perez, A. Gonzalez-Prieto, J. Diaz, D. Lopez-Fernandez, J. Garcia-Martin, and A. Yague, “Devops research-based teaching using qualitative research and inter-coder agreement,” *IEEE Transactions on Software Engineering*, 2021.
- [67] A. Abirami, S. Pudumalar *et al.*, “Active learning strategies and blended learning approach for teaching under graduate software engineering course,” *Journal of Engineering Education Transformations*, vol. 35, no. 1, pp. 42–51, 2021.
- [68] J. S. Okolica, A. C. Lin, and G. L. Peterson, “Gaming devsecops-a serious game pilot study,” in *National Cyber Summit*. Springer, 2020, pp. 61–77.
- [69] H. Christensen, “Crunch: Automated assessment of microservice architecture assignments with formative feedback,” in *European Conference on Software Architecture*. Springer, 2018, pp. 175–190.
- [70] K. Kuusinen and S. Albertsen, “Industry-academy collaboration in teaching devops and continuous delivery to software engineering students: towards improved industrial relevance in higher education,” in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 2019, pp. 23–27.
- [71] S. Krusche and L. Alperowitz, “Introduction of continuous delivery in multi-customer project courses,” in *Companion Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 335–343.
- [72] H. Christensen, “Teaching devops and cloud computing using a cognitive apprenticeship and story-telling approach,” in *Proceedings of the 2016 ACM conference on innovation and technology in computer science education*, 2016, pp. 174–179.
- [73] E. Bobrov, A. Bucchiarone, A. Capozucca, N. Guelfi, M. Mazzara, and S. Masyagin, “Teaching devops in academia and industry: reflections and vision,” in *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer, 2019, pp. 1–14.
- [74] I. Alves and C. Rocha, “Qualifying software engineers undergraduates in devops-challenges of introducing technical and non-technical concepts in a project-oriented course,” in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 2021, pp. 144–153.
- [75] S. Mirhosseini and C. Parmin, “Opunit: Sanity checks for computing environments,” in *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer, 2019, pp. 167–180.
- [76] J. Kousa, P. Ihantola, A. Hellas, and M. Luukkainen, “Teaching container-based devops practices,” in *International Conference on Web Engineering*. Springer, 2020, pp. 494–502.
- [77] A. Rahman, S. Hossain, and D. B. Bose, “Exercise perceptions: Experience report from a secure software development course,” in *International Conference on the Quality of Information and Communications Technology*. Springer, 2021, pp. 521–535.
- [78] M. Hills, “Introducing devops techniques in a software construction class,” in *2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 2020, pp. 1–5.
- [79] H. Golafrooz Shahri and M. Khaghanizade, “Introduction to oral presentation teaching method,” *Education Strategies in Medical Sciences*, vol. 2, no. 4, pp. 161–166, 2010.
- [80] R. Hobeck, I. Weber, L. Bass, and H. Yasar, “Teaching devops: a tale of two universities,” in *Proceedings of the 2021 ACM SIGPLAN International Symposium on SPLASH-E*, 2021, pp. 26–31.
- [81] M. K. Rao, “Storytelling and puzzles in a software engineering course,” *ACM SIGCSE Bulletin*, vol. 38, no. 1, pp. 418–422, 2006.
- [82] S. Ouhbi and M. A. M. Awad, “The impact of combining storytelling with lecture on female students in software engineering education,” in *2021 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 2021, pp. 443–447.
- [83] B. Wei, Y. Li, L. Deng, and N. Visalli, “Teaching distributed software architecture by building an industrial level e-commerce application,” in *International Conference on Software Engineering Research, Management and Applications*. Springer, 2019, pp. 43–54.
- [84] M. Mazzara, A. Naumchev, L. Safina, A. Sillitti, and K. Urysov, “Teaching devops in corporate environments,” in *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer, 2018, pp. 100–111.
- [85] E. Larios Vargas, M. Aniche, C. Treude, M. Bruntink, and G. Gousios, “Selecting third-party libraries: The practitioners’ perspective,” in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 245–256.
- [86] E. G. Guba, “Criteria for assessing the trustworthiness of naturalistic inquiries,” *Ectj*, vol. 29, no. 2, pp. 75–91, 1981.
- [87] I. Korstjens and A. Moser, “Series: Practical guidance to qualitative research. part 4: Trustworthiness and publishing,” *European Journal of General Practice*, vol. 24, no. 1, pp. 120–124, 2018.