From Theory to Interpreting the Practice: Exploring the Role-play to Teach DevOps

Lucas Lima Mota
Federal University Ceará
Crateús, CE, Brazil
lucaslmota@alu.ufc.br

Rodrigo Pereira dos Santos
Federal University of the State of Rio de Janeiro
Rio de Janeiro, RJ, Brazil
rps@uniriotec.br

Awdren Fontão
Federal University of Mato Grosso do Sul
Campo Grande, MS, Brazil
awdren.fontao@ufms.br

Allysson Allex Araújo
Federal University of Cariri
Juazeiro do Norte, Ceará, Brazil
allysson.araujo@ufca.edu.br

ABSTRACT

[Context] While Role-play (RP) has recently attracted increased attention due its capability as an active learning method in Software Engineering (SE) education, a research gap remains regarding its adoption within the context of DevOps, a critical area in SE. [Objective] This study investigates a RP-based teaching model for DevOps. [Method] We follow an empirical, experimental, and descriptive research approach, employing structured questionnaires and participant observation based on three RP sessions (n=30 students). Data analysis included Descriptive Statistics and Thematic Content Analysis. [Results] In summary, 90% of the students agreed that RP, when compared to an usual DevOps class, was more captivating. Moreover, 93.3% of them agreed that RP contributed to their learning. Pair-based learning and integration of industry tools were highly favored by students according to qualitative results. [Contributions] We offer an RP-based teaching model that encourages collaboration and technical proficiency in SE students. Our proposal can also serve as a potential source of inspiration for companies designing in-company training processes.

CCS CONCEPTS

• **Social and professional topics → Software engineering education**.

KEYWORDS

Role-play, Active Learning, DevOps.

1 INTRODUCTION

DevOps has become a fundamental topic for both research and industry in the field of Software Engineering (SE) [22]. In general, DevOps is a organizational movement that aims to orchestrate the development (Dev) and operations (Ops) processes to facilitate system development and deployment [23]. While the implementation of DevOps may vary from one organization to another, its core principles are based on three pillars: Continuous Integration (CI), Continuous Delivery (CDE), and Continuous Deployment (CD) [18, 33, 39]. From the educational perspective, recent efforts by SE scholars [2, 9, 32] have been made towards developing instructional material for DevOps education in line with industry demands.

Given this context, there is a recognized need to engage in discussions about efficient pedagogical approaches for teaching DevOps, addressing specific challenges such as conceptual complexity, the difficulty of replicating real-world environments, and limited educational resources [7, 24, 36]. This issue aligns with the premise that SE education should encompass skills and attitudes essential for the job market while striving for educational excellence [26].

According to Bonwell and Eison [5], active learning methods are fundamental to engage students and enhance skills based on action-reflection-action. In particular, Role-play (RP) has emerged as an active method that contributes to learning and interaction among students [1]. Rabelo and Garcia [35] explain that RP promotes to the students act in specific roles within a given context, leveraging the construction of knowledge through critical reflection. In the context of SE education, RP has been prominently explored due to its ability to offer an experience close to the "reality" of the job market and promote active learning [20, 42]. However, despite the relevance of using RP for educational purposes, there is a gap in the SE literature regarding the study of teaching approaches based on RP for specific DevOps scenarios.

Considering the motivation outlined, this study is justified by the importance of sharing experiences regarding using RP as a pedagogical tool to enhance DevOps education. We justify the relevance of investigating the RP due its capability of encouraging active student participation and, consequently, increase engagement and motivation to learn [8, 13]. Therefore, this research adopts an experimental and mixed-method scope to address the following research question: "*What are the experiences and perceptions of undergraduate students regarding the adoption of RP for learning DevOps?*". To answer this question, we initially propose a RP-based teaching model and subsequently discuss an empirical evaluation involving three experimental sessions (totaling 30 students).

Concerning the academic contributions, our study offers a generic teaching model based on RP that can be used by educators and students within the DevOps education context. By assuming specific roles, engaging in a simulated context, and dealing with day-to-day problems and decisions, students can apply the theoretical concepts learned in the classroom practically and reflectively. This educational approach helps students develop technical knowledge and essential skills such as teamwork, communication, problem-solving, and decision-making.

Regarding the contribution to practice/industry, adopting RP as a teaching strategy for DevOps contributes to developing more prepared professionals capable of addressing industry challenges. By experiencing situations analogous to those faced in a work environment, students build competencies and gain practical experience in applying DevOps practices and tools. Additionally, leveraging RP encourages student interaction and collaboration, fostering collective knowledge construction and refining interpersonal skills highly valued in the software industry. Our proposal can also serve as inspiration for companies designing in-company training processes (*e.g.*, intern onboarding) following RP.

This paper is organized as follows. Section 2 presents our related work. Section 3 clarifies our research method. Section 5 analyzes our results, while Section 6 presents a general discussion of our findings. Section 7 addresses the limitations. Finally, Section 8 approaches our concluding remarks.

2 RELATED WORK

This section presents empirical and related works that address the use of RP in SE education. Dixon and Jagodzinski [14], for example, conducted a qualitative analysis of RP exercises using an ethnographic approach based on the thinking aloud method, commonly employed in simulated case studies within application domains. The analytical approach adopted in the study revolved around a thinking-aloud exercise followed by qualitative data analysis. This practice involved a participant (an expert in the domain) assuming a role and verbalizing their thoughts aloud while performing the exercise. The session was recorded and transcribed for subsequent analysis. The authors concluded that thinking aloud RP exercises made visible the user's work practices, revealing previously hidden details about what was being done and how it was being executed.

Henry and LaFrance [20] introduced five RP exercises applicable within a SE course. These exercises encompassed guidelines to conduct RP sessions and a set of debriefing questions. Students were assigned roles of presenter and reviewer, with project groups rotating to present segments of their work to other groups. Subsequently, each participating student completed a questionnaire, yielding 175 responses. 96% of respondents assessed the exercises as effectively utilizing class time, while 98% advocated for their inclusion in future courses, and 99% deemed them relevant to the course material. Additionally, 89% reported achieving the intended learning outcomes through the exercises. Only 5% expressed a preference for traditional lectures over the exercises. These findings emphasized the efficacy of RP as a versatile tool capable of fostering applications within SE education, with nearly all students perceiving it as a valuable learning aid.

Diaz Redondo et al. [13] reported on an experiment integrating collaborative learning and RP strategies with Web 2.0 in SE. The proposed activity prioritized the development of social and design competencies, which are essential for software development success. Collaborative learning was integrated with ongoing assessment, under a student-centered pedagogy. The authors adopted a project-based approach and an immersive pedagogical strategy incorporating RP. The experiments spanned three years, during which several positive outcomes were observed. Students exhibited heightened motivation to tackle real-world problems within a gaming context. Furthermore, the Web 2.0 emerged as a learning and appealing component to telecommunications engineering students interested in emerging technologies.

Decker and Simkins [12] employed RP in a course on Game Development Processes. The authors devised an RP framework tailored to the requirements of the course, structured as follows: 1) In the first week, students were introduced to the overarching narrative for the semester; 2) The second week involved an exploration of emerging trends in the gaming industry; 3) Week three featured a brainstorming session aimed at fostering student ideation for projects; 4) Week four emphasized commercial considerations; 5) Week five delved into ongoing projects within the industry; 6) During the sixth and seventh weeks, students were tasked with presenting software development descriptions; 7) In the tenth week, topics and readings pertaining to legal issues and risk analysis were covered; 8) The thirteenth and fourteenth weeks concluded with final presentations of semester-long projects. Informal inquiries were conducted at the end of the semester to collect feedback on the impact of RP scenarios on course topic comprehension and learning. Students provided predominantly positive opinions on the inclusion of RP in the curriculum.

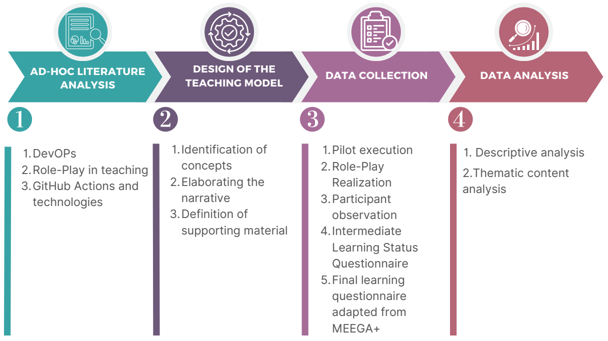
Maxim et al. [27] addressed RP activities to simulate the work experience in a professional game development studio to enhance an advanced undergraduate game design course. Alongside RP, a gamification framework was devised and integrated into the course to enable students to personalize their engagement. The project execution involved weekly three-hour meetings over 14 weeks. Subsequently, students undertook the first task, which involved crafting a game design document and a business plan for the game. The second deliverable encompassed an alpha prototype of the game, comprising a complete logical path, a draft user document, and a functional installer. The third deliverable was a beta prototype necessitating accommodation of a requirement change, and the final product was the launch prototype alongside a marketing presentation, including a promotional video for the game product. Course evaluation was ongoing, with students slated to complete an assessment form subsequently.

As discussed earlier, there have been studies that use Role-Playing (RP) to teach Software Engineering (SE). However, there are currently no studies that use RP to teach DevOps, despite its significant role in both academia and practice. Hence, this study aims to explore the use of RP as an active learning method to provide experiences that align with the demands of the software industry in the context of DevOps concepts and practices.

3 RESEARCH METHOD

The methodological trajectory pursued in this study follows a quali-quantitative approach, aiming to empirically comprehend undergraduate students' experiences and perceptions regarding the adoption of RP for DevOps learning. Furthermore, this study is experimental and descriptive, with data collection guided by structured questionnaires and participant observation. Descriptive statistics and Thematic Content Analysis were utilized for data analysis. Adhering to data condensation conventions, data display, and the design/verification of conclusions proposed by Miles et al. [28], this research adopts a multimethod scope organized into four macro-stages: 1) ad-hoc literature analysis, 2) teaching model conception, 3) data collection, and 4) data analysis. These methodological procedures can be seen in Figure 1.

Figure 1: Methodological procedures.


AD-HOC LITERATURE ANALYSIS
DESIGN OF THE TEACHING MODEL
DATA COLLECTION
DATA ANALYSIS
1
1. DevOPs
2. Role-Play in teaching
3. GitHub Actions and technologies
2
1. Identification of concepts
2. Elaborating the narrative
3. Definition of supporting material
3
1. Pilot execution
2. Role-Play Realization
3. Participant observation
4. Intermediate Learning Status Questionnaire
5. Final learning questionnaire adapted from MEEGA+
4
1. Descriptive analysis
2. Thematic content analysis

Initially, an **ad-hoc analysis of the literature** was carried out, aiming to achieve a comprehensive understanding of the theoretical and practical components relevant to the proposal. This bibliographic analysis spanned from September 2021 to March 2022 and involved manual searches on Google and Google Scholar using combinations of keywords pertinent to the research domain (Role-play, DevOps, Continuous Integration, etc.). Our focus was primarily on peer-reviewed articles from conferences and scientific journals, as well as undergraduate theses, written in both Portuguese and English. Notably, two seminal works were identified during this phase. Firstly, Leite et al. [25]'s survey provided an exhaustive mapping of foundational DevOps concepts. Secondly, Cherif and Somervill [8]'s research offered a set of guidelines for conducting RP activities in educational settings.

After comprehending the fundamental concepts required for this study, it became possible to begin the macro-stage of **developing an teaching model** based on RP. This model drew inspiration from the directives delineated by Cherif and Somervill [8]. Regarding didactic content, the decision was made to underpin the proposal based on the survey crafted by Leite et al. [25], thereby aiding in the identification of pertinent categories, tools, and concepts in DevOps. Leveraging the gleaned information, tailored for this study, a narrative comprising three phases was devised, covering the potential onboarding process of junior software developers in a hypothetical small-scale software development company.

Concerning the **data collection**, it occurred both *during* and *after* the RP. Participants (students acting as software developers), guided by the facilitator (first author of this paper representing the Tech Lead in the company), engaged in the outlined script within the RP-based teaching model. This teaching model was evaluated through an educational experience of three sessions (T1, T2, T3) with students from different semesters of the Computer Science and Information Systems undergraduate programs. We employed a purposeful sampling approach to target students approved in the Software Engineering discipline. In addition, participants had to be currently enrolled in the university's undergraduate degree program, as our sessions were conducted in-person in a university laboratory. We recruited students through in-person invitations, WhatsApp messages in student groups, and snowball sampling (participants inviting their close colleagues). Two pilot executions were also carried out to validate the procedures adopted, such as data collection instruments, PR dynamics, etc.

After validating the procedures, sessions for assessing RP were conducted separately with three distinct sessions in a laboratory setting at the Federal University of Ceará (UFC) - Crateús Campus, on November 1st, 16th, and 29th, 2023, respectively. During these sessions, students were paired up. T1 comprised individuals with the highest perceived knowledge of DevOps, with 90% reporting levels 7, 8, or 9. Additionally, this cohort exhibited a predominantly male composition (90%). T2, conversely, displayed a more balanced scenario in terms of both knowledge levels and gender distribution, with a slight female majority (56%). Lastly, T3 represented the cohort with the highest proportion of individuals declaring level 1 knowledge (45.45%). There was a predominance of male participants (73%) in T3. For more detailed participant characterization data, please refer to the supporting repository [29].

For data collection, two distinct techniques were employed: structured questionnaires and participant observation. Regarding structured questionnaires, data collection occurred at two different moments using two distinct questionnaires. Initially, periodically after the completion of each phase of RP, serving as an intermediary status covering the participant's experience rather than solely at the end. For these moments, the first questionnaire was administered, comprising two questions: 1) Based on your understanding of the concepts addressed in this phase, please rate your level of comprehension from 1 to 5, with 1 being the lowest level and 5 the highest level; 2) Based on your experience during this phase, do you have any open-ended comments you would like to share? This approach enabled us making specific adjustments to each phase of the model based on the obtained responses in the future.

The second questionnaire was administered after the completion of the entire experimental process, i.e., after the student had traversed all phases of the RP-based session. To this end, we designed a questionnaire inspired by the MEEGA+ model [40]. We slightly adapted the questions to explicitly mention the RP. MEEGA+ was chosen because it provides a structured framework that closely aligns with the objective of our research, specifically in effectively evaluating the 'player experience' [31]. Moreover, MEEGA+ has already demonstrated extensive validity in prior educational research within the SE [17, 38], ensuring reliability and relevance. As such, the use of MEEGA+ allowed us to rigorously assess the impact of RP activities on student learning experience in DevOps education.

Another approach employed for data collection was through participant observation. This method was enacted during the onboarding scenario of junior software developers within the fictitious company, which was designed for the RP. Consequently, interactions were structured to occur between pairs of students or among other pairs, enabling participants to verbalize their prior or acquired knowledge during the RP as needed. To ensure that interactions unfolded optimally, the facilitator, in the guise of a Tech Lead, mediated some exchanges by addressing specific doubts from students and providing fundamental knowledge for RP execution.These interactions resulted in material that was collected after the conclusion of each section, which was made up of feedbacks that varied from the participants' verbal and social expressions.

Finally, regarding **data analysis**, it was observed that deliberations regarding the questionnaires, in addition to the data obtained through participant observation, constitute a rich multiplicity of discussions. Descriptive statistics were employed for the analysis of quantitative results, while qualitative data underwent Thematic Content Analysis [3, 11]. Inspired by the works of Braun and Clarke [6] and Holton [21], a four-step qualitative analysis procedure was established. In the first step, an assimilation with the data was sought during the transcription of the conducted interviews and through the reading and re-reading of the notes. Subsequently, the process advanced with *open coding*, aimed at identifying related utterances and observations, organizing them according to the addressed theme. Following this, in the third step, *axial coding* was conducted to link the initially emerged utterances and themes, with the purpose of identifying sub-themes and creating categories that exhibit similarity, facilitating the amalgamation of results. This process was initially conducted dynamically and inductively by the author and later refined by the advisors (facilitating interpretative comparison). Finally, a comprehensive evaluation of the analyzed information was performed to materialize the findings (with the codebook available in our repository [29], thereby ensuring transparency and accessibility in regard to our research.

4 D&O: A RP-BASED TEACHING MODEL

The teaching model based on Role-Play (RP) proposed in this study, named *Dev & Ops* - **D&O** (a title inspired by the renowned RPG Dungeons and Dragons - D&D), was initially influenced by the guidelines set forth by Cherif and Somervill [8]. According to their research, there are five essential elements for conducting RP: 1) formulation of the problem or topic to be addressed; 2) characters or roles to be performed; 3) context and information that students must know or research; 4) roles or stages that each student or group of students must fulfill; and 5) procedures for presenting the dramatization of the piece. In this context, D&O also draws upon Bloom's Taxonomy [4] concerning educational objectives, which are divided into three domains: cognitive, affective, and psychomotor. In light of these pedagogical directions, the operation of D&O is elucidated as follows.

Regarding the **problem formulation**, we need to provide students with conditions to learn, from a theoretical-practical perspective, the basic DevOps. To define which concepts would be explored (either conceptually or practically), this study adopted the survey mapped by Leite et al. [25]. Due to time constraints and to avoid excessive complexity, some concepts were superficially mentioned, while others were explained in greater theoretical detail and subsequently practiced. However, we aimed to cover the majority of concepts per category in terms of practices. Table 1 indicates whether a specific concept was briefly mentioned, explained, or practiced during the dramatization. Then, a narrative structure was established based on the onboarding process of newly hired junior software engineers at a small software development company that adopts Scrum and DevOps practices. A Tech Lead welcomes this team and explains the development pipeline adopted in the organization and the requirements requested by the Product Owner (PO). This demand entails building a NodeJS-based Application Programming Interface (API) that encompasses three Functional Requirements (FRs): FR1) create a user, FR2) query a list of users, and FR3) query a specific user.

Table 1: Approached DevOps concepts from Leite et al. [25]

Category	Concept	Mentioned	Explained	Practiced
Knowledge Sharing	Culture of collaboration		✓	✓
	Knowledge sharing		✓	✓
	Breaking down silos	✓		
	Cooperation between departments	✓		
Source code management	Version control		✓	✓
	Culture of collaboration		✓	✓
	Sharing knowledge		✓	✓
	Breaking down silos	✓		
	Collaboration between departments	✓		
Building process	Release engineering		✓	✓
	Continuous delivery	✓		
	Automation	✓		
	Test automation		✓	✓
	Static analysis		✓	✓
Continuous integration	Frequent and reliable launch process		✓	✓
	Release engineering	✓		
	Continuous integration		✓	✓
	Deployment pipeline		✓	✓
	Continuous delivery and automation	✓		
	Artifact management		✓	✓
Deployment automation	Frequent and reliable launch process	✓		
	Release engineering	✓		
	Configuration management		✓	✓
	Continuous deployment		✓	✓
	Infrastructure as code	✓		
	Virtualization and containerization	✓		
	Cloud services and automation	✓		
Monitoring and registration	You built it, you run it		✓	✓
	After-hours support for devs	✓		
	Continuous runtime monitoring	✓		
	Performance, availability and scalability	✓		
	Resilience, reliability and automation	✓		
	Metrics, alerts and experiments		✓	✓
	Log management and security		✓	✓

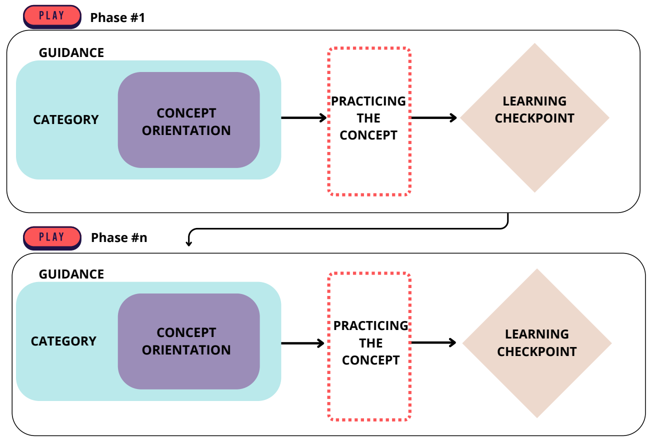
Regarding the **roles or characters to be played in the scenario**, we have established three possible options inspired on Scrum guidelines. The first role, known as the Tech Lead, will be responsible for guiding the entire scenario from both a pedagogical and technical perspective. We recommend that this role be assumed by the professor or teaching assistant of the course. The second role will be that of the Product Owner, which will not be played by any participant but will represent a Non-Player Character (NPC) to demonstrate the influence of the PO in a software project context. Finally, we have the Dev Team, which will be portrayed by a team of students who will practice the concepts and practices guided by the Tech Lead.

As for the **context and information that students should know or research**, the premise was that students already have knowledge of programming web and SE, being those who have completed the course or already approved, for example. We may expect that students have a prior foundation on the pillars of the software development process in general, in particular, knowledge of the concepts around building an API, knowledge of the Representational State Transfer (REST), Node Package Manager (NPM), and basic knowledge of Git and GitHub. However, the dynamics designed here aim to specifically deepen the concepts of DevOps. Therefore, students can previously and briefly know the basic elements of that area, but it is not mandatory.

Regarding **the roles that students must fulfill**, there is the premise that the Dev Team (formed by students) will be responsible for participating in onboard in the company and, consequently, building the demand requested by the PO in a manner that adheres to the development pipeline adopted in the organization to which they were recently hired. In the context of a class with many students, we suggest that they be organized into pairs by computer, similar to the practice of pair programming.

Finally, the **procedures for presenting the play dramatization** are detailed below. Initially, the first phase focused on knowledge sharing and source code management. Subsequently, in the second phase, the core of the dramatization revolved around the building process, continuous integration, and deployment automation. Lastly, the emphasis shifted to monitoring and logging in the third phase. The suggested and established script for the dramatization (with a duration of up to 100 minutes) aims to be pedagogically simple to replicate yet robust enough to cover the key DevOps concepts outlined by Leite et al. [25]. However, it is important to note that, being an RP-based proposal, there is room for possible adaptations and adjustments, either beforehand in terms of planning by the instructor or even improvisationally during the dynamics as the dramatization progresses. Hence, the phases of the script serve as an instrumental framework to guide the dynamics both technically and narratively. In addition to the full narrative script, this work also provides a slide presentation to guide the Tech Lead. These materials are open available in our repository [29].

Figure 2: Overview of the D&O Teaching Model.



The model illustrated in Figure 2, drawing inspiration from microlearning [19], comprises three distinct phases, each aimed at being objectively concise to mitigate learner fatigue. During the *Concept Orientation* phase, the Tech Lead takes charge and provides a clear understanding to the Development Team regarding what will be explored in the respective phase. This phase is analogous to onboarding new employees in a real-world software development environment. The second phase, termed *Practicing the Concept*, aims to provide the Dev Team with opportunities to practically apply the previously discussed concepts. This exercise should assist learners in technically addressing the Product Owner's demands and in cultivating soft skills. The third and final phase entails a *Learning Checkpoint* conducted with the students to assess their overall understanding of the specific phase. This checkpoint involves verifying, with the students, the fulfillment of specific phase objectives.

With the intention of ensuring that the entire dynamic proceeds within the allotted time frame and achieves the objectives of each phase, interventions during the dynamic process are permitted. Interventions may occur within the same team, where participants with more experience immediately assist their peers. If challenges persist, other pairs that are making progress with their tasks may support participants from different pairs, aiming to uphold the established time constraints. If there is no progress in achieving the objectives within or between teams, the Tech Lead should intervene immediately, demonstrating to everyone the resolution of the proposed activities.

5 RESULTS

5.1 Lessons from pilot sessions

Before starting the activity with the three groups, pilot executions were decided to validate the adopted procedures. The purpose was to validate aspects such as the duration of the activity and data collection instruments, gather feedback, and identify any potential need for changes in supporting slides. Initially, during the first pilot conducted on October 13, 2023, it was observed that the time required for the activity was excessively long due to the configuration of the machines used and the excessive amount of dialogue from the Tech Lead role (the pilot lasted approximately 120 minutes). This issue impacted, for example, the time available to complete the questionnaires. Consequently, adjustments were made to the slides used to conduct the activity, and efforts were made to configure the development environment in the laboratory. After these adjustments, a second pilot with a different pair of students was conducted on October 17, 2023. The second pilot ensured that the objectives were achieved, maintaining an execution time within the stipulated timeframe (100 minutes), allowing for the completion of data collection forms. Additionally, based on student feedback, we noticed sufficient learning of the RP execution.

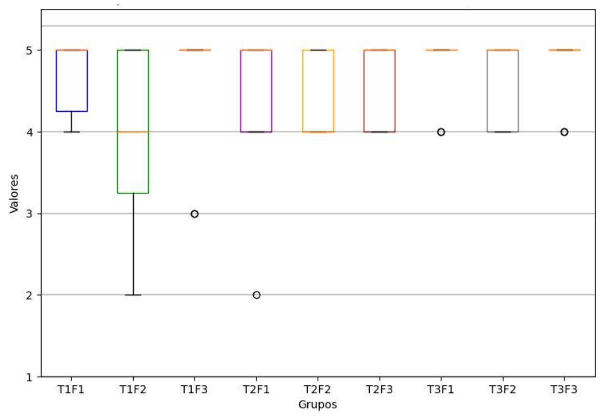
5.2 Individual phase analysis

In this section, we will discuss the results of the questionnaires that were administered in between the phases. The analysis of the questionnaires was based on two questions: 1) Based on your understanding of the DevOps concepts covered in this phase of the PR dynamics, please rate your level of understanding from 1 to 5, with 1 being the most low and 5 the highest level and 2) Based on your experience during the level, are there any open comments you want to make?

5.2.1 Quantitative assessment of the level of understanding of the concepts of each phase. The results obtained from each session (T1, T2, and T3) for the three phases of the RP (F1, F2, and F3) are presented in Figure 3. For instance, T1F1 refers to the results obtained by the T1 session based on the concepts learned in F1.

In T1, which had 10 participants, we observed that 70% of the students had a comprehension level of 4-5 for F1. The median score for comprehension in this phase was 5, with the first quartile at 4 and the third quartile at 5. In F2, the proportion of students reporting their comprehension level remained consistent, with a median score of 4, the first quartile at 3, and the third quartile at 5. In F3, 80% of the students reported a high comprehension level of 5, resulting in a median score of 5, which was consistent with both the first and third quartiles.

Figure 3: Level of understanding at each stage per session.



Concerning T2, which had 9 participants, we found that 88.8% of the students reported their understanding level between 4 and 5 for F1, with the median being 5. The first quartile was 4, and the third quartile was 5. For F2 and F3 phases, we observed that all students (100%) mentioned their understanding level between 4 and 5. The median for F2 was 4, with the first quartile also being 4 and the third quartile being 5. For F3, the first quartile was 4, the median was 5, and the third quartile was 5.

In relation to T3, which had 11 participants, all students showed a comprehension level between 4 and 5 for F1, resulting in a median of 5. The first and second quartiles were also equal to the median. The comprehension levels for both F2 and F3 were the same, with F2 showing a median of 5, the first quartile at 4, and the third quartile at 5. Similarly, F3 had median values of 5, with both the first and third quartiles aligning with the median.

In conclusion, the results showed considerable progress in the students' understanding of the subject, with an overall increase in class scores across the three phases. T1 students demonstrated a significant improvement in their comprehension level, with 80% indicating the highest level of knowledge in the final phase. T2 and T3 cohorts consistently showed an improvement in their understanding, with the majority of students rating their comprehension level between 4 and 5 across all phases. These findings highlight the effectiveness of the RP-based approach in teaching DevOps concepts over the phases.

5.2.2 *Qualitative assessment of the experience during each phase.*

The purpose of this section is to analyze the students' experiences in each phase of the RP dynamics. This analysis is based on the responses gathered from the open-ended question in the questionnaire administered in the end of each phase. We summarize in Table 2 our codebook comprising the emerged themes, codes, and associated participants.

Table 2: Codebook regarding the experience in each phase.

Theme	Codes	Participants
Experiences at each stage	Cooperative Learning	P8,P10,P18,P22,P23,P26,P27,P29
	Learning with Industry Tools	P4,P5,P8,P19,P12,P28,P29
	Challenges and Positive Points	P4,P15,P14,P23,P27,P20,P22

Regarding the theme of **Experiences in each phase**, three distinct codes were identified. The first code, termed Cooperative Learning, pertains to participants' feedback regarding the opportunity to explore cooperative practices as a didactic mechanism, exemplified by the utilization of pair programming. Participants elaborated on their impressions concerning the theoretical and practical contents of the RP. Regarding Phase 1, P8 elucidated: "*The dynamics of pair programming were highly productive as they underscored the importance of cooperation and communication among team members, as well as across the entire organization*". P10 remarked: "*It was enlightening to learn that DevOps commences with a more collaborative approach*". Concerning Phase 2, P18 commented: "*It was intriguing to witness that prior to delving into code, there was a moment dedicated to demonstrating teamwork and knowledge sharing*". In addition, P23 expressed: "*The discussed topic was intriguing, especially considering its limited discourse within the university setting. Moreover, the methodology aids in learning in a clear and cohesive manner*".

One of the objectives of the dynamic was to provide participants with an immersion of concepts and practices that are similar to what the industry expects, but without forgetting the importance of theoretical foundations. In this sense, the code Learning with Industry Tools shows students' receptivity to the use of practices that are addressed in the industry and the tools linked to them. Regarding F1, P4 highlighted that: "*It was interesting to see in practice the process that is applied in large companies in a more summarized and quick way, thus further strengthening the concepts*". For P5: "*It was interesting to put into practice the first steps to upload a project*". P19 highlighted: "*Interesting attention given to the naming standard related to Git. This makes it easier to view the project, along with its history*". P28, when learning about versioning, reported that: "*There was good use in understanding and practicing versioning, contributing to knowledge in DevOps*".

During F2 there was a greater awareness of the use of tools such as Git, Fly.io and CI/CD pipelines for CI/CD. According to P8: "*It was very interesting to upload the application to Git and see the entire integration working as expected, giving an error when the tests did not pass and being approved when the tests were successfully passed*". P12 was surprised to use a hosting tool, he said: "*It was very interesting, I didn't know the Fly.io process. However, with the instructions it was clear*". P19, in your analysis, said that: "*Release engineering is something very important within a software project. It is more organized and its activities are specific; also helping to release more versions for production, so that updates are not lost in the process and making it easier to visualize the code*".

In reference to F3, P29 articulated that: "*The establishment and implementation of automated pipelines to comprehend the entirety of the formatting, testing, and deployment workflow within our applications constitute a highly advantageous practice, one that possesses the potential to significantly benefit the development team as a cohesive unit, provided it is effectively leveraged*". P8 also underscored the relevance of employing monitoring tools, asserting: "*The utilization of static analysis via SonarQube proved noteworthy, emerging as an indispensable instrument for fostering optimal software development practices*". On the practice of *logs*, P19 reflected: "*It's something very important among the test development team, and it was very important to address it. It's useful in several situations, but one of them* *is when the system has a problem or the server goes down, the logs help to map where the problem came from, with precise information, which also helps in testing, such as date, time, machine, OS, etc.*".

The Challenges and Positive Points emerged as another code bringing an overview of the challenges and positive aspects associated with our use of RP. Concerning the experience with F1, P15 remarked: "*The initial nomenclatures were somewhat complex, but I appreciated the didactic approach*". Regarding F2, P4 noted, "*More complex, yet very comprehensive from the perspective of the entire process*". Conversely, P14 highlighted the presence of: "*Many details that can easily lead to errors if the step-by-step process is not followed correctly*". In elucidating their stance, P27 articulated: "*I appreciated the theoretical framework alongside its practical application, as it aids in solidifying understanding*".

Regarding F3, P4 remarked that: "*Despite a few errors, the process was highly satisfactory and dynamic, and the learning experience throughout all phases until deployment was commendable*". Agreeing with P4, P20 expressed the desire for "*More activities in this direction*". Echoing these sentiments, similar to the previous statement, P22 said the following: "*I believe that the practice was relatively simple because of previous knowledge, in general, I thought it was super cool*". P27, for her part, said: "*I found it a very good and dynamic way of understanding the content*".

Upon examination of the aforementioned data, a discernible trend emerged regarding the integration of cooperative learning, utilization of industry tools for learning purposes, and the identification of both challenges and merits. Specifically, it becomes evident that students appreciated the opportunity to engage in various practices inherent to RP, encompassing pair programming and the requisite tools for DevOps implementation.

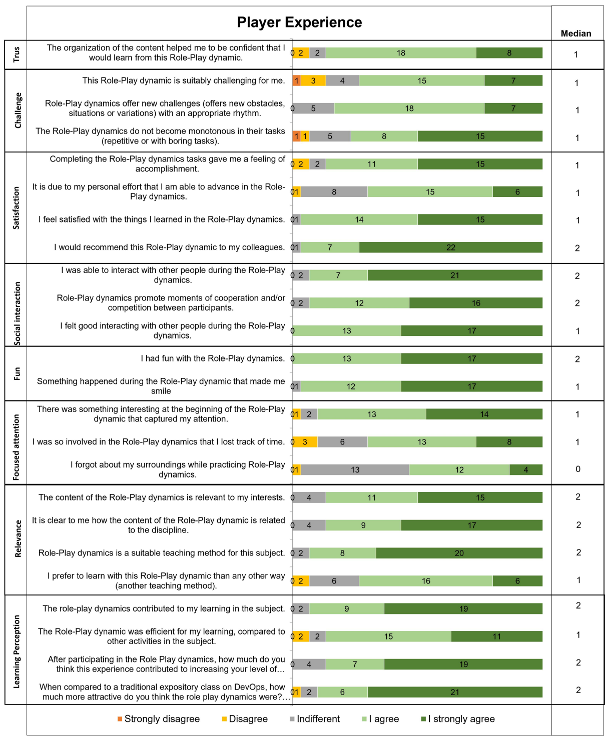
5.3 Analysis based on the MEEGA+

This study delved into the MEEGA+ model for player experience evaluation [30]. The questionnaire originally proposed within the MEEGA+ framework encompasses two distinct perspectives: usability and player experience. In alignment with the objectives of this study, our evaluation opted to concentrate on the player experience perspective, since the elements addressed in usability are not relevant to the study's objective. Regarding the player experience, the assessment covered the following dimensions: Confidence, Challenge, Satisfaction, Social Interaction, Enjoyment, Focused Attention, Relevance, and Learning Perception. These dimensions are discussed individually below, while Figure 4 provides an overview of the obtained results.

Regarding the data concerning **Trust**, approximately 86.6% of the participants agreed that the content organization increased their confidence in their learning outcomes. In our approach, we organized the RP following a set of slides covering the resources in a linear structure within a temporal chronology. Another artifact that contributed to enhanced trust among students was the utilization of a wiki, enabling them to access instructions for the practice's.

The data regarding the **Challenge** revealed that 73.3% of students agreed that the dynamics presented a satisfactorily challenging level, while 83.3% also expressed agreement regarding the introduction of new challenges at an appropriate pace. Additionally, 76.6% of students agreed that the dynamics successfully avoided monotony. This category exhibited promising outcomes, as the challenges were designed to mitigate doubts and hesitation during executions. There was a blend of programming, package management, DevOps pipelines, and application monitoring.

Figure 4: Results from the player experience [40].



Regarding **Satisfaction**, we observed that 86.6% of the students agreed in expressing satisfaction upon completing the tasks of the activity. Furthermore, 70% agreed in their ability to progress in the activities through personal effort. 96.6% of the participants agreed regarding their satisfaction with the acquired learning. Lastly, 73% of students agreed that they would recommend the activity to their peers. These favorable outcomes reflect the structured content of the RP, which aimed to facilitate task completion for all students without undue strain, encouraging them to devise solutions for programmed scenarios, including the resolution of minor errors.

Upon analyzing the data concerning to **Social Interaction**, we observed that approximately 70% of the participants expressed agreement regarding the feasibility of interaction throughout the dynamic session. Furthermore, 93.3% of the participants agreed that the activity fostered cooperative situations, while 100% asserted a positive sense of engagement when interacting with fellow participants. These outcomes can be attributed to the progression of pair programming as the software requirements were being formulated, thereby encouraging collaboration and knowledge sharing among students. This approach facilitated sustained interaction among all participants alongside all dynamic session.

The results pertinent to **Entertainment** indicate that 100% of the students agreed that they enjoyed the RP dynamics. This result was quite positive. Moreover, 96.6% of the students agreed that certain situations during the dynamics elicited laughter. Regarding this aspect, there was no deliberate inclusion of comedic relief in the design of the activity; rather, it arose naturally through the positive engagement among participants.

Concerning the **Focused Attention**, we noticed that 90% of the participants agreed that there were engaging elements at the onset of the RP that captured their attention. Additionally, 70% of them agreed to being so immersed in the RP that they lost track of time, while 83.3% reported agreeing that they forgot the environment they were immersed in during the RP. Different factors may elucidate these findings. Initially, the number of participants warrants consideration, as each RP session involved an average of 10 participants, amidst distractions, conversations, and interventions during the practices, thereby potentially hindering a state of focused concentration. Furthermore, participants' fatigue from daily commitments could be a contributing factor, as some students exhibited signs of weariness.

By analyzing **Relevance**, we may verify that 86.6% of the respondents agreed that the content addressed in the RP was important. Additionally, 86.6% acknowledged that the RP content is directly related to the discipline. Furthermore, 93.3% affirmed that the RP constitutes a suitable method for the discipline. Lastly, 73.3% of students expressed a preference for learning through the RP compared to other methods. These results underscore the RP's prominence as an active learning method, affording participants a practical learning experience distinct from passive roles assumed in traditional lectures. Another aspect highlighted by participants during the dynamic pertains to the opportunity for hands-on experience, wherein students conceive, publish, and monitor project executions. This aspect received considerable acclaim from the students.

About the **Learning Perception**, we observed that 93.3% of the participants agreed that RP contributed to their learning. Additionally, 86.6% of them agreed that RP was effective in learning, and 86.6% agreed that RP was a direct factor contributing to knowledge aggregation. Lastly, 90% of the students agreed that RP, when compared to a traditional DevOps class, was more captivating. These findings are related to different factors that promoted an engaging learning environment. For example, we carefully selected theoretical and practical concepts to enhance students' knowledge. As a result, we designed an RP session that facilitated knowledge acquisition and enriched the students' perception of their learning experience.

The analysis of player experience data from MEEGA+ revealed consistently positive outcomes across all dimensions. Responses indicating neutrality or indifference were more prevalent regarding the sensation of forgetting the surrounding environment during RP practice. This observation may be attributed to factors such as session size and duration, acknowledging the challenge of sustaining student engagement throughout extended sessions.

5.4 Analysis of the experience as a whole

This section focuses on the analysis of responses to an open-ended question regarding students' overall experience with RP, as derived from the questionnaire administered at the end of the dynamic. Two distinct themes and five codes emerged, as synthesized in Table 3.

Table 3: Codebook regarding the experience as a whole.

Theme	Codes	Participants
Positive Aspects	Exploring Practices and Industry Tools	P1,P2,P6,P14,P28,P30
	Pair-based Learning	P8, P11
	RP Acceptance	P6,P12,P13,P16,P15,18,P24,P28
Improvement Perspectives	Time Management	P14,P15,P16,P19
	RP's Dynamics Enhancements	P13,P19,P20,P28

Concerning the initial theme, **Positive Aspects**, three distinct codes were observed. The first code, Exploring Practices and Industry Tools, reinforced the students' positive perception regarding the utilization of tools in the learning process. For example, P2 enjoyed "*using SonarQube to detect Code Smells*". In turn, P6 emphasized that "*the integration with Fly, a novel tool I learned and can potentially use in future projects. Its complex configuration may pose a challenge, but for small-scale applications, its utilization becomes quite straightforward. Additionally, the aspect of interaction with logs was fundamental for understanding how the production linkage with another tool that retrieves these files is accomplished*". This practical engagement with the DevOps concepts was also perceived as a positive factor by P14: "*The hands-on approach is essential for better solidifying the learning*". Moreover, P28 underscored: "*The distinct interactivity compared to conventional classroom settings*". As one can see, students expressed enthusiasm for adopting modern software development practices like CI/CD and industry tools. They appreciated using SonarQube for detecting potential code issues, indicating a focus on software quality. The integration with Fly.io was also positively received, showing its potential for future projects. Therefore, the "hands-on" experience inherent to the RP sessions, in contrast to traditional classroom settings, deserves recognition.

Another code referred to the role of Pair-based Learning to promote learning. Inspired by pair programming, our RP proposal facilitated knowledge exchange among students, including error handling and attention to the software quality. This practice was highly valued by students. For instance, P8 and P11 emphasized the importance of correct use of pair programming and constant collaboration between the pair. More specifically, P8 said: "*What I liked most was the choice of pair programming, my conclusion was that this choice was necessary for the good assimilation of knowledge and to force the student to participate and interact as much as possible*". P31 also enjoyed "*the division of roles and the constant contribution with my partner*".

Regarding RP Acceptance, we observed positive feedback towards the DevOps learning experience and an easily understandable approach to complex topics, making the content comprehensible. This perspective was evident in the speech of P6: "*Well-expository and easily understandable, this practice could simplify complex topics and their peculiarities*". In accordance with P6, P16 also stated: "*I quite liked the practice itself, for acquiring new insights*"". P12 and P18 also expressed interest in the DevOps, respectively stating: "*I found the covered content very interesting, it explained a lot about DevOps to me*"; and "*Knowledge of DevOps is extremely important*". P15 noted: "*Overall, the RP was an alternative and enjoyable experience*". P28 said: "*The presented dynamic as well as the topics covered were of great assistance and brought knowledge about important technologies and tools*". In conclusion, these responses indicate a strong acceptance of the RP, particularly in its ability to convey complex concepts effectively and provide valuable lessons into DevOps practices.

The second theme delves into codes derived from **Improvement Perspectives**, focusing on aspects that could enhance the dynamics of the RP. One prominent code within this theme is Time Management. Different respondents expressed a concern about the required time for the practice, advocating for dedicated sessions for each phase to facilitate deeper assimilation. For instance, P14 emphasized the need for more time, stating, "*More time, too much information in a short span*". This sentiment was echoed by P16, who advocated for "*More time and dedicated sessions for each phase*". Another suggestion aimed to streamline practical sessions by excluding purely theoretical concepts, as suggested by P19: "*Perhaps if some purely theoretical parts were omitted to expedite the process*". Additionally, there was contention regarding the complexity of the second phase, which consumed a large portion of the allotted timeframe, as highlighted by P15: "*The time between phases. The second phase was more complex and consumed a substantial portion of time*". In summary, these speeches collectively address the need for a balanced and adaptable approach to optimize learning during RP session, evidencing the importance of managing time effectively to ensure a productive learning experience.

Finally, the code RP's Dynamics Enhancements aims to reflect potential improvements in the RP's dynamics as a whole. One initial observation came from P20, who suggested adjusting the steps to better anticipate common errors in user installation and related processes. Similarly, P13 emphasized the importance of clarifying the context by providing functional and operational examples during implementation to enhance understanding. Furthermore, P19 proposed introducing the technologies utilized in the examples and practices to enrich the learning experience. P28 elucidated that: "*Adding additional contributors would facilitate the comprehensive utilization of all roles, such as the Non-Player Character (NPC)*". These speeches highlight the need for clear contextualization, practical examples, and comprehensive role utilization to enhance the operationality and functionality of the RP's dynamics.

6 DISCUSSION AND LESSONS LEARNED

This study investigates the adoption of Role-Play (RP) as an active learning method for DevOps education. The research question guiding this investigation is: *"What are the experiences and perceptions of undergraduate students regarding the adoption of Role-Play for learning DevOps?"*. We developed a RP-based teaching model to address this question and conducted three RP sessions involving 30 students. Our data collection methods included questionnaires and participant observation, while data analysis involved descriptive statistics and thematic content analysis. This mixed-method approach allowed us to investigate important experiences and perceptions arising from the instructional practice.

DevOps has been in high demand in SE education due to its widespread adoption in the software industry [16, 22]. More recently, Ferino and Kulesza [15] found that educators should favor practical methods in a collaborative learning environment to teach DevOps. Our study's results are consistent with previous research about RP in SE education, showing that students were receptive to using RP. Our RP-based teaching sessions aimed to provide a balanced mix of theoretical and practical content. As a result, we emphasized the importance of giving students a solid understanding of DevOps practices. To achieve this objective, we embraced the idea that SE education should be more student-centered, with the goal of increasing their motivation, participation, and ultimately, their learning [10, 34]. In our experience, 90% of students found RP more engaging than traditional DevOps classes, and 93.3% agreed that RP contributed to their learning.

In regard to the lessons learned, we found that students greatly valued the hands-on exploration of industry practices and tools, such as continuous integration, version control, static code analysis, and release engineering. It is worth noting that this goes beyond simply introducing students to a set of tools; it offers them practical experience with a set of principles meant to foster an efficient team [37]. Additionally, pair-based learning facilitated effective collaboration among students. Furthermore, RP received positive feedback for its capability to approach complex topics in an enjoyable experience. These lessons are particularly important since we are pioneering the use of RP in the context of DevOps, whose syllabus presents different challenges.

In addition, our qualitative analysis revealed two major issues to improve from the students' perspective. Firstly, some respondents expressed concerns about the required time for the practice, advocating for dedicated sessions for each phase to facilitate deeper assimilation. Secondly, some students suggested resizing the amount of theoretical content and mentioned a lack of familiarity with certain theoretical and practical concepts, such as setting up DevOps pipelines and deploying applications created by the students. They also noticed that the second phase was considered more complex than the others. To address these challenges, we consider restructuring the sessions to allow more focused time for each phase and adjusting the theoretical content to ensure it is more manageable.

In the viewpoint of the instructor, incorporating a mix of visual aids (the slides deck), hands-on activities, and cooperative learning allowed for addressing various learning preferences. Of course, for any pedagogical approach even nowadays, it is a challenge to unify and necessarily deliver the same level of learning for students with different backgrounds. However, we provided support and flexibility within each session, which was organized into three distinct phases. For instance, students were encouraged to collaborate in pairs during hands-on exercises, allowing them to learn from each other's strengths and perspectives. Additionally, we implemented individual assessments after each phase, coined as Learning Checkpoints, and adjusted the pace as necessary.

In summary, this research explores an immersive teaching method integrating theoretical and practical DevOps concepts through an RP-based model. Designed to simulate professional environments, this model enhances student engagement and fosters the development of technical and interpersonal skills. From an industry standpoint, Kuusinen and Albertsen [24] and Sánchez-Cifo et al. [37] highlight a global shortage of developers and a growing gap between industry demand and educational supply, largely due to misalignment between education systems and market needs. Therefore, we emphasize the importance of experiential learning opportunities that reflect real-world challenges, bridging the gap between academic training and industry requirements.

7 THREATS TO VALIDITY

This section examines the threats to validity [41] that influenced our study and how we mitigated them. One threat to **external validity** relates to the sample size, which consisted of 30 undergraduate students in Computer Science and Information Systems. Although the sample size provided pertinent results, expanding and diversifying the sample is important to enhance generalizability. Nevertheless, it is reasonable to argue that the number of participants reached, spread across three distinct cohorts with diverse profiles, contributed to achieving our intended research objective. To mitigate threats to **construct validity**, we ensured that participants had a basic understanding of the investigated theme. Familiarity with some aspects of the theme helped address potential misalignments in basic knowledge. Before introducing the RP's dynamics, we provided detailed explanations of the proposed method, associated concepts, and practices, ensuring that all participants had a clear understanding of the study's context.

Regarding **internal validity,** we tried to maintain an engaging environment during data collection. In this regard, participants completed two different questionnaires: one between phases and another at the end of the session. The final questionnaire was based on the MEEGA+, which has already demonstrated extensive validity in SE education. Additionally, there was variability in participants' knowledge levels despite the mandatory completion of the SE course. To address this issue, we organized students into pairs, inspired by pair-programming practices, to facilitate mutual assistance. The facilitator was available to intervene when necessary, and participants had access to a supportive wiki for additional resources. In terms of **conclusion validity**, potential behavioral issues such as introversion or fear of judgment could undermine the effectiveness of the approach. We observed that reinforcing participant presentations and engaging in informal dialogues helped mitigate these concerns, promoting a more inclusive and collaborative environment. To ensure our experience can be replicated, we provided a set of slides and a description of the RP phases in our Zenodo repository [29].

8 CONCLUSION

The study focused on exploring undergraduate students' perceptions and experiences regarding the utilization of Role-Play (RP) for learning DevOps. We followed a structured approach consisting of four methodological steps. Firstly, an analysis of existing literature was conducted to gain a deep understanding of the key connections between DevOps and RP within the context of Software Engineering (SE) education. Secondly, a RP-based teaching model was designed, which included definitions such as concept identification, crafting narratives, and defining supporting materials. Thirdly, data collection occurred through interim and final questionnaires to enable an in-depth exploration of RP experience. Finally, Descriptive Statistical Analysis and Thematic Content Analysis were carried out to analyze and interpret the data.

Our proposed RP-based teaching model, named D&O, underwent evaluation through an educational assessment comprising three sessions with 30 participants from Computer Science and Information Systems undergraduate programs. Quantitative evaluation of comprehension levels (on a scale of 1 to 5) showed that 93.3% of students across the cohorts reported a high level of comprehension (4 and 5) for each phase of the model. In addition, open-ended feedback revealed overall positive acceptance, particularly emphasizing cooperative learning aligned with industry-explored processes and technologies. Analysis of categories from the MEEGA+ model indicated agreement on RP's contribution to learning (93.3%), its captivation compared to traditional DevOps classes (90%), fostering cooperative situations (93.3%), and generating a positive sense of engagement (100%) among participants. Feedback also highlighted positive aspects, such as pair-based learning, with potential areas for improvement like time management.

Regarding academic contributions, our study offers a generic teaching model based on RP that can benefit SE educators and students within the DevOps education context and beyond. This approach facilitates the practical application of theoretical concepts by assuming specific roles, engaging in a simulated context, and addressing day-to-day challenges. Our research indicates that utilizing RP as an active learning method helps students to develop technical knowledge while also cultivating valuable soft skills.

In the context of practice and industry, adopting RP as a teaching strategy for DevOps contributes to preparing professionals better equipped to tackle real-world industry challenges. Additionally, leveraging RP encourages student interaction and collaboration, emphasizing a holistic approach to SE education that includes technical proficiency and soft skills refinement. This proposal also has the potential to inspire companies in designing in-company training processes, such as intern onboarding, following RP principles.

Future research could explore adapting the D&O model beyond DevOps to areas like software testing and requirements engineering. Another direction is modifying the content for asynchronous study, with Tech Lead and Product Owner roles facilitated by generative AI. Additionally, assessing the proposal from an instructor's perspective within a specific research protocol would be valuable.

ARTIFACTS AVAILABILITY

All data supporting this study are openly available through our supporting repository [29].

ACKNOWLEDGEMENTS

The second author thanks CNPq (Proc. 316510/2023-8), FAPERJ (Proc. 211.583/2019), CAPES and UNIRIO (PPQ) for partial support.

REFERENCES

[1] Marvin C Alkin and Christina A Christie. 2002. The use of role-play in teaching evaluation. *American Journal of Evaluation* 23, 2 (2002), 209–218.

[2] Isaque Alves and Carla Rocha. 2021. 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, 144–153.

[3] Laurence Bardin. 1979. Análise de conteúdo. *Lisboa: Edições* (1979).

[4] Benjamin S Bloom, David R Krathwohl, Bertram B Masia, et al. 1984. Bloom taxonomy of educational objectives. In *Allyn and Bacon.* Pearson Education London.

[5] Charles C Bonwell and James A Eison. 1991. *Active learning: Creating excitement in the classroom. 1991 ASHE-ERIC higher education reports.* ERIC.

[6] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. Qualitative research in psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101.

[7] Alfredo Capozucca, Nicolas Guelfi, and Benoît Ries. 2019. Design of a (yet another?) devops course. In *Software Engineering Aspects of Continuous Development*

and New Paradigms of Software Production and Deployment: First International Workshop, DEVOPS 2018, Chateau de Villebrumier, France, March 5-6, 2018, Revised Selected Papers 1. Springer, 1–18.

[8] Abour H Cherif and Christine H Somervill. 1995. Maximizing learning: using role playing in the classroom. *The American Biology Teacher* (1995), 28–33.

[9] Henrik Bærbak Christensen. 2016. 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*. 174–179.

[10] Orges Cico, Letizia Jaccheri, Anh Nguyen-Duc, and He Zhang. 2021. Exploring the intersection between software industry and Software Engineering education - A systematic mapping of Software Engineering Trends. *Journal of Systems and Software* 172 (2021), 110736. https://doi.org/10.1016/j.jss.2020.110736

[11] Victoria Clarke and Virginia Braun. 2017. Thematic analysis. *The journal of positive psychology* 12, 3 (2017), 297–298.

[12] Adrienne Decker and David Simkins. 2016. Leveraging role play to explore software and game development process. In *2016 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–5.

[13] Rebeca P Diaz Redondo, Ana Fernandez Vilas, Jose J Pazos Arias, and Alberto Gil Solla. 2014. Collaborative and role-play strategies in software engineering learning with web 2.0 tools. *Computer applications in engineering education* 22, 4 (2014), 658–668.

[14] A Dixon and AP Jagodzinski. 2003. Qualitative data analysis of thinking-aloud role-play exercises: Usefulness in software-engineering requirements analysis. *Concurrent Engineering: Advanced Design, Production and Management Systems* 20 (2003), 107–113.

[15] Samuel Ferino and Uirá Kulesza. 2023. Unveiling the Teaching Methods Adopted in DevOps Courses. In *Proceedings of the XXII Brazilian Symposium on Software Quality*. 355–357.

[16] Marcelo Fernandes, Samuel Ferino, Uirá Kulesza, and Eduardo Aranha. 2020. Challenges and recommendations in devops education: A systematic literature review. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*. 648–657.

[17] Thaís de Souza Deluca Ferreira, Davi Viana, and Rodrigo Pereira dos Santos. 2021. Árvore de ecos: Um jogo para ensino de conceitos de ecossistemas de software. *Revista Brasileira de Informática na Educação* 29 (2021), 273–300.

[18] Martin Fowler. 2006. Continuous Integration. https://www.martinfowler.com/articles/continuousIntegration.html Acessado em: 19/11/2021.

[19] Silvia Gabrielli, Stephen Kimani, and Tiziana Catarci. 2017. The design of microlearning experiences: A research agenda (on microlearning). (2017).

[20] Tyson R Henry and Janine LaFrance. 2006. Integrating role-play into software engineering courses. *Journal of Computing Sciences in Colleges* 22, 2 (2006), 32–38.

[21] Judith A Holton. 2007. The coding process and its challenges. *The Sage handbook of grounded theory* 3 (2007), 265–289.

[22] Ramtin Jabbari, Nauman bin Ali, Kai Petersen, and Binish Tanveer. 2016. What is DevOps? A systematic mapping study on definitions and practices. In *Proceedings of the scientific workshop proceedings of XP2016*. 1–11.

[23] Gene Kim, Kevin Behr, and Kim Spafford. 2014. *The phoenix project: A novel about IT, DevOps, and helping your business win*. IT Revolution.

[24] Kati Kuusinen and Sofus Albertsen. 2019. 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, 23–27.

[25] Leonardo Leite, Carla Rocha, Fabio Kon, Dejan Milojicic, and Paulo Meirelles. 2019. A survey of DevOps concepts and challenges. *ACM Computing Surveys (CSUR)* 52, 6 (2019), 1–35.

[26] José Vinícius Vieira Lima, Cleverton Anderson Duarte Silva, Fernanda Maria Ribeiro de Alencar, and Wylliams Barbosa Santos. 2020. Metodologias Ativas como forma de reduzir os desafios do ensino em Engenharia de Software: diagnóstico de um survey. In *Anais do XXXI Simpósio Brasileiro de Informática na Educaçao*. SBC, 172–181.

[27] Bruce R Maxim, Stein Brunvand, and Adrienne Decker. 2017. Use of role-play and gamification in a software project course. In *2017 IEEE frontiers in education conference (FIE)*. IEEE, 1–5.

[28] Matthew B Miles, A Michael Huberman, Michael A Huberman, and Michael Huberman. 1994. *Qualitative data analysis: An expanded sourcebook*. sage.

[29] Lucas Lima Mota, Rodrigo Pereira dos Santos, Awdren Fontão, and Allysson Allex Araújo. 2024. From Theory to Interpreting the Practice: Exploring the Role-play to Teach DevOps - Supporting repository. https://zenodo.org/records/11222118

[30] Giani Petri, Christiane Gresse von Wangenheim, and Adriano Ferreti Borgatto. 2019. MEEGA+: Um Modelo para a Avaliação de Jogos Educacionais para o ensino de Computação. *Revista Brasileira de Informática na Educação* 27, 3 (2019).

[31] Giani Petri, Christiane Gresse von Wangenheim, and Adriano Ferreti Borgatto. 2017. Evolução de um Modelo de Avaliação de Jogos para o Ensino de Computação. In *Anais do XXV Workshop sobre Educação em Computação*. SBC.

[32] Anne-Marie Pinna-Déry. 2019. Teaching DevOps at the Graduate Level. In *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment: First International Workshop, DEVOPS 2018, Chateau de Villebrumier, France, March 5-6, 2018, Revised Selected Papers*, Vol. 11350. Springer, 60.

[33] Sten Pittet. 2021. Integração contínua vs. entrega contínua vs. implantação contínua. https://www.atlassian.com/br/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment Acessado em: 19/11/2021.

[34] Rafael Prikladnicki, Adriano Bessa Albuquerque, Christiane G von Wangenheim, and Reinaldo Cabral. 2009. Ensino de engenharia de software: desafios, estratégias de ensino e lições aprendidas. *FEES-Fórum de Educação em Engenharia de Software* (2009), 1–8.

[35] Lísia Rabelo and Vera Lúcia Garcia. 2015. Role-play para o desenvolvimento de habilidades de comunicação e relacionais. *Revista Brasileira de Educação Médica* 39, 4 (2015), 586–596.

[36] Guoping Rong, Shenghui Gu, He Zhang, and Dong Shao. 2017. DevOpsEnvy: an education support system for DevOps. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 37–46.

[37] Miguel Ángel Sánchez-Cifo, Pablo Bermejo, and Elena Navarro. 2023. DevOps: Is there a gap between education and industry? *Journal of Software: Evolution and Process* 35, 12 (2023), e2534.

[38] Rodrigo Ribeiro Silva, Luis Rivero, and Rodrigo Pereira Dos Santos. 2021. Programse: Um jogo para aprendizagem de conceitos de lógica de programação. *Revista Brasileira de Informática na Educação* 29 (2021), 301–330.

[39] Unity. 2023. O que é CI/CD? https://unity.com/pt/solutions/what-is-ci-cd Acessado em: 04/03/2023.

[40] C Wangenheim, G Hauck, J Boigatto, F Adriano, and LHM Pacheco. 2018. MEEGA+ A model for evaluating educational games. http://www.gqs.ufsc.br/quality-evaluation/meega-plus/ Acessado em: 24/05/2023.

[41] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, Anders Wesslén, et al. 2012. *Experimentation in software engineering*. Vol. 236. Springer.

[42] Didar Zowghi and Suresh Paryani. 2003. Teaching requirements engineering through role playing: Lessons learnt. In *Proceedings. 11th IEEE International Requirements Engineering Conference, 2003.* IEEE, 233–241.