A Pedagogical Architecture for Systems Design and Development in Remote Learning Based on Software Ecosystems Characteristics

Rodrigo Feitosa
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
rfeitosa@cos.ufrj.br

Paulo Malcher
Federal Rural University of Amazônia
Capitão Poço, Brazil
paulo.malcher@ufra.edu.br

Davi Viana
Federal University of Maranhão
São Luís, Brazil
davi.viana@ufma.br

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

ABSTRACT

The COVID-19 pandemic forced universities to transition to remote learning, presenting challenges for teaching programming. Thus, it is important to teach programming through new system development trends, such as software ecosystems (SECO), that consider social and technical aspects to bring students closer to the reality of the industry, even remotely. This work aims to propose and apply PA4SDD-SECO, a pedagogical architecture for systems design and development in remote learning based on SECO characteristics. PA4SDD-SECO was applied in an undergraduate information systems course, where an observational study was used to analyze the relationship of influence among students, and a questionnaire was used to analyze both PA4SDD-SECO and the relationship of influence among students according to their perceptions. As a result, PA4SDD-SECO improved student cooperation, regardless of geographic distance, and helped share experiences. We found that feedback directed at one project in SECO can influence others, even when geographically distant.

KEYWORDS

Systems Design and Development, Pedagogical Architecture, Software Ecosystems, Influence, Software Quality Education

1 Introduction

The systems design and development have their content distributed in several disciplines related to Software Engineering (SE) in the computing courses (e.g., computer science and information systems). However, the experience of isolated disciplines makes it difficult for students to develop a systemic view of the software development process [64]. Therefore, computing courses also explore disciplines that put all this content into practice. These disciplines allow the observation of both the technical aspects of system development and the social nature of developers' interactions. Software ecosystems (SECO) consider technical and social aspects of software development and can bring students closer to the reality of the industry [59], particularly in managing software quality aspects such as modularity, reuse, and collaborative improvement of evolving systems. SECO are the interaction of software and actors (i.e., keystone, end-users, third-party developers) about a common technological platform, resulting in a set of contributions and directly or indirectly influencing the ecosystem [45].

The influence is an important social aspect in software development, which is explored in SECO [21]. It is "*a type of power, on which having power over someone is to make someone else do something they would not do otherwise*" [69]. In the context of SE education, which is a discipline of a practical nature, we can observe the student's influence on others in the performance of their tasks in teamwork [2], potentially impacting software quality attributes such as consistency, correctness, and maintainability of shared deliverables. In programming, for example, students attend computing courses with distinct levels of experience, which can influence the development and the quality of the software product [19, 71]. Hence, it is necessary to encourage relationships between students using knowledge from the current market that favors this interaction, such as SECO.

Several researchers have searched for innovative teaching methods for higher education to encourage student interactions [4, 28, 39, 49]. A common point highlighted by these researchers was the need for teaching approaches that allowed interaction/socialization and cooperative/collaborative learning among those involved. This interaction can be challenging to construct knowledge, mainly when it encompasses students, teaching assistants, and professors physically separated in time and/or space [29, 50, 60, 74]. Challenges related to being physically separated have increased with the emergence of COVID-19 and spreading across the globe [5]. Online participation and computational resources to deliver courses replaced physical presence. Given this context, research on pedagogical approaches aimed at teaching systems design and development in remote teaching and learning has been the object of study [25, 63].

This work proposes a pedagogical architecture (PA) for systems design and development in remote learning based on SECO characteristics called PA4SDD-SECO. PA can be defined as learning structures that combine different components articulated from pedagogical, technological, and political approaches [14, 24]. PA are approaches that favor using technologies within an ecosystemic vision [55]. We apply the PA proposed in this paper in the discipline of Systems Design and Development offered at a university in an undergraduate course on Information Systems. The course involves

the development of software projects considering quality aspects such as requirements, development process, iterations, deadlines, deliveries, and so on. We analyze their application from an observational study that also investigated the relationship of influence between students and the characteristics of an influencer in SECO. In addition, we applied a questionnaire to analyze PA4SDD-SECO and the relationship of influence between students according to their perceptions.

As a result, the proposed PA contributed meaningfully to the remote learning context of the Systems Design and Development discipline. Our findings suggest that PA4SDD-SECO fosters student cooperation regardless of geographic dispersion, promoting effective knowledge sharing and exchange of experiences. Notably, we observed an influential dynamic between novice and more experienced students, particularly during the early stages of the projects, which enhanced collaborative learning. Furthermore, the impact of directed feedback extended beyond individual teams, influencing the quality and direction of specific projects and generating ripple effects across the entire ecosystem, reinforcing a culture of collective improvement and continuous learning.

The remainder of this paper is organized as follows: Section 2 presents the background and related work; Section 3 explains the proposed PA; Section 4 describes the evaluation method; Section 5 presents the results; Section 6 discusses the findings; finally, Section 7 concludes the work with final remarks and future work.

2 Background and Related Work

This study discusses the changes in teaching SE content, including the system's design and development. SE teaching is based on systematically applying scientific and technological knowledge through methods and experience in project construction, system implementation, software testing, and documentation [10, 40]. In SE teaching, projects are complex tasks based on challenging questions or problems that engage students in design, problem-solving, and decision making activities [62]. This requires methods or approaches that can support this context of teaching.

Computing courses are adopting innovative teaching methods to improve SE education and students' skills throughout their academic experience in the system's design and development [22]. The use of student-centered practices such as hackathons [70], crowdsourcing [32], and teaching agile software development [9] has been used. Moreover, novel ways to develop systems with solid support for appropriate SE techniques have emerged using these innovative methods, such as SECO [45]. SECO can be defined as groups of projects that are developed and co-evolve in the same environment [44].

The literature highlights a gap between methods and approaches that use SECO characteristics to support SE teaching, although it is an interesting complement [15, 16]. Approaches that use SECO characteristics to help practice in computing education can provide a more global view of relationships between suppliers and customers around a common technological platform [16, 51]. According to Lettner et al. [41], there is a growing agreement about the following SECO's general characteristics: internal and external developers; common technological platform; keystone; platform enabling external contributions; variability-enabled architectures; shared core assets; tools, frameworks and patterns; and distribution channel. Thus, the global view that SECO characteristics enable in system development can assist student interaction.

The use of technologies in education expands interactive spaces for the shared construction of knowledge among actors at several times and without the need for professors and students to share the same geographic spaces [68]. According to Aivaloglou and Meulen [2], team and individual assignments in computing education have characteristics that can influence how students approach and experience their team/individual responsibility in a distinct geographic space. This view among computing students can be gleaned through social influence theory. Kelman [35] defines three types of influence: (1) compliance: is a type of power that refers to the act of responding to an explicit or implicit request from the third party to achieve a specific goal, such as winning an award, approval, or avoiding punishment; (2) identification: is the change in attitudes or behaviors due to the influence of someone who is admired; and (3) internalization: is the process of accepting a set of norms established by people or groups that influence the individual and the individual accepts it because the content of the accepted influence is intrinsically rewarding.

Farias et al. [21] investigated the definition and characteristics of influencers in SECO. The authors defined influencers as actors that can begin leading development and dictate how the software project will progress in a SECO and extracted eight relevant characteristics related to social interaction or the contribution of technical code (Table 1). In turn, Condina et al. [12] conducted an exploratory study on the sense of the influence of developers in an open source SECO. The authors recognized aspects that reinforce some of the characteristics of an influencer defined by Farias et al. [21].

Table 1: Characteristics of influencers in [21].

Characteristic	Category
Source of learning	Technical
Participation with code	Technical
Content value	Technical
Long-time interaction with the project	Social
Participation with comments	Social
Closeness to the GitHub project owner	Social
Status in the project	Social
Status (popularity on GitHub)	Social

Portilho et al. [54] presented a PA for teaching programming logic to high school students, showing gains in engagement and learning through digital and collaborative strategies. Similarly, Lima and Menezes [42] proposed a framework for introductory programming based on problem-solving and constructivist principles to foster autonomy and active learning. In the context of Requirements Engineering education, Santana et al. [18] proposed a PA that integrates practical activities and digital tools to develop hard and soft skills and strengthen the connection between academic learning and professional practice.

Unlike the works mentioned above, this work presents a PA for systems design and development in remote learning based on SECO characteristics. It also analyzes the relationship of influence

between students through the characteristics of an influencer in SECO during the proposed PA application.

3 Proposed Pedagogical Architecture

PA4SDD-SECO considers three components, according to Tavares et al. [43]: pedagogical conception; educational strategy; and computational support. Figure 1 illustrates the structure of the PA. This structure is formed by the components of PA4SDD-SECO and the SECO characteristics on which it is based, as presented next.

3.1 Pedagogical Conception

The pedagogical conception of PA4SDD-SECO is based on Piaget's cognitive development model [52], which emphasizes cooperative work (in teams) from the introduction to programming. Working in teams fosters the development of formal reasoning, stimulates students' metacognition by encouraging them to articulate and defend their thoughts, and enhances learning through the exchange of perspectives on the subject matter.

In addition to this cognitive foundation, the pedagogical conception incorporates SECO characteristics, as identified by Lettner et al. [41], to promote collaboration, reuse, and co-evolution of software artifacts within an educational context. This approach also fosters software quality attributes such as modularity, traceability, and maintainability through structured interactions among teams. These characteristics guided the design of PA4SDD-SECO and are detailed as follows:

- **Internal and external developers:** In SECO, development is no longer restricted to an internal team, but it is open to external developers. We promoted interaction between students on the same team (internal developers) and students of different teams (external developers), allowing them to act as business partners. This allowed for fostering cooperation and relationship of influence between teams, including tasks such as bug fixing, suggestion of new features, and expansion of projects across teams;
- **Common technological platform:** SECO relies on a common technological platform to support a specific technology and a class of applications. We defined a common technological platform in which teams can cooperate, follow, and access each other's projects, regardless of geographical location due to remote learning;
- **Keystone:** In SECO, a keystone is typically responsible for providing and controlling the common technological platform. In PA4SDD-SECO, the professor and teaching assistants are assigned the responsibility of defining part of the computational support, tracking the progress of teams' projects, and providing feedback on each iteration of systems design and development;
- **Platform enabling external contributions:** The common technological platform has to be designed and implemented to enable external contributions in SECO. We allowed teams to copy existing repositories in order to create their own ones. The teams could send merge requests to the original repositories, helping other team members review and incorporate their contributions;
- **Variability-enabled architectures:** The common technological platform of a SECO has to ease the development of a set of applications customized to the customers' specific needs. We allowed the teams to define the most viable architecture for their projects, enabling them to incorporate software artifacts that could be adapted to different contexts, environments, or the specific purposes of their projects;
- **Shared core assets:** SECO typically use reusable features and components. We defined that the common technological platform and virtual learning environment (VLE) would be used as repositories for code, documentation, and presentations with access available to all teams;
- **Tools, frameworks and patterns:** SECO further relies on tools, frameworks, and patterns to manage the development and evolution of the variability-enabled architecture, the common technological platform, and the shared core assets and contributions. We defined some software for the students to model and prototype their systems. We also allowed each team to define the tools, structures, and standards to support the development and evolution of their project over the common technological platform;
- **Distribution channel:** A specific characteristic of several SECO is the existence of a freely accessible distribution channel for third-party development. We defined a set of channels for distributing and sharing information that allows the maintenance of communication between teams, professor, and teaching assistants.

3.2 Educational Strategy

The educational strategy of PA4SDD-SECO emphasizes the importance of each participant's engagement and contribution to teaching and learning. Thus, the stages of this strategy were defined to foster cooperation between teams working on software projects, considering quality aspects such as requirements, development processes, iterations, deadlines, and deliveries.

I. **Presentation of syllabus and project definition:** The professor presents the guiding elements for the system's design and development in the discipline. Students are grouped into teams and define their projects;
II. **Review on systems design:** The professor revised concepts and practices centered on systems design to the level of everyone's theoretical knowledge;
III. **Presentation of computational support:** The professor and teaching assistants present SECO characteristics, such as the common technological platform, and organize projects to co-evolve in this environment. Students select technologies and tools of their choice to achieve their project goals;
IV. **System development and delivery:** Teams perform systems design and development over the common technological platform. The professor and teaching assistants guide students in executing projects synchronously and asynchronously throughout the process. System development and delivery stage is organized by iteration (each 15 days);
V. **Final assessment of projects:** The professor and teaching assistants receive the projects. The assessment process is based on the quality of artifacts produced by teams in each

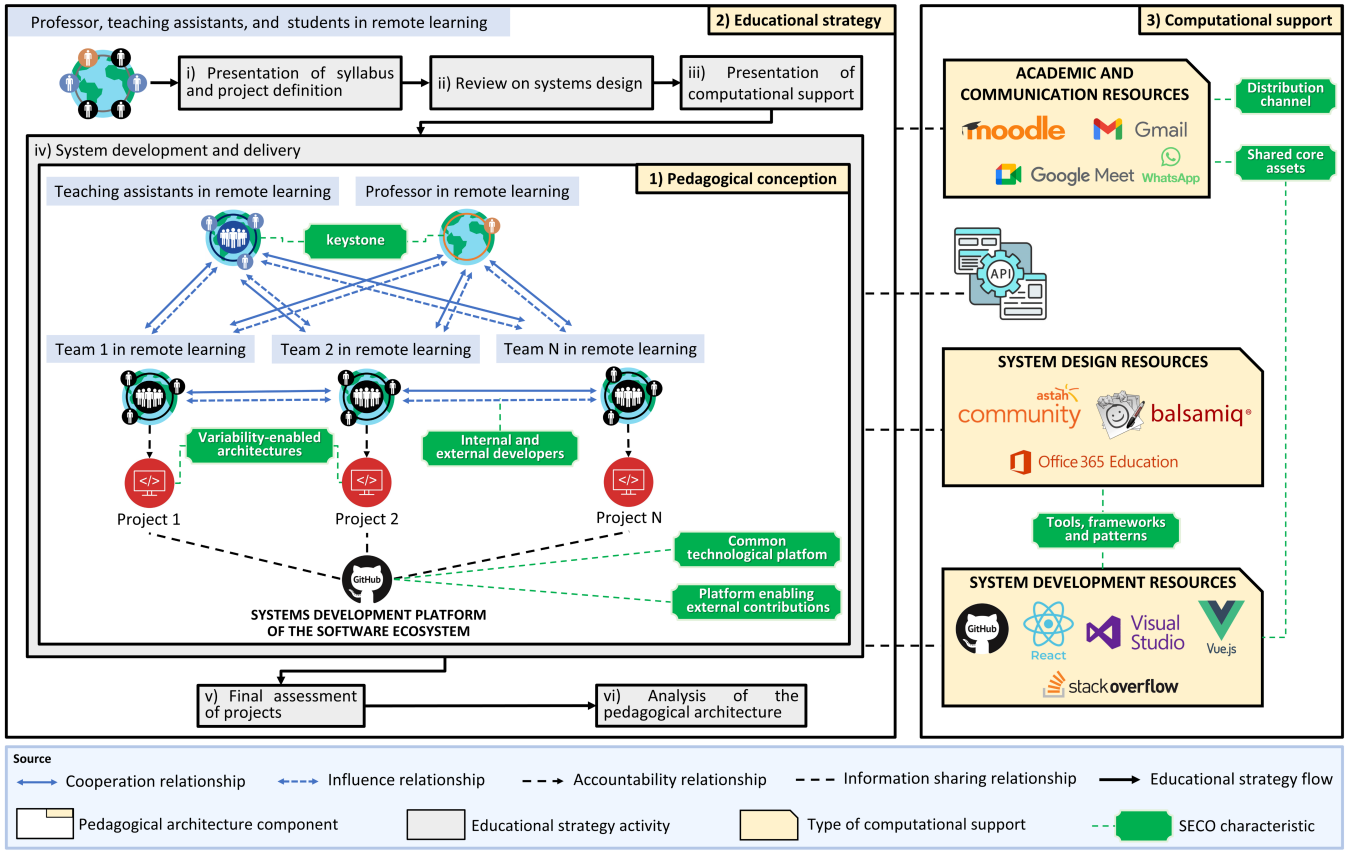

Figure 1: PA4SDD-SECO - PA for systems design and development in remote learning based on SECO characteristics.

iteration, their ability to achieve interaction and performance goals, and participation in project activities;

VI. **Analysis of the PA:** Students analyze aspects related to the application of PA4SDD-SECO and provide suggestions for its improvement.

3.3 Computational Support

The computational support facilitates cooperation, information sharing, and knowledge exchange and is divided into the following three groups of computational resources.

- **Academic and communication resources** that facilitate remote communication between professor, students, and teaching assistants: (i) ***Moodle*** - tool used to provide materials, send reminders, and define, monitor, and evaluate tasks; (ii) ***Google Meet*** - tool used for synchronous remote classes; (iii) ***Gmail*** - tool used to create email threads to centralize communication with each team; and (iv) ***WhatsApp*** - messaging application used for quick communication, coordination of activities, and real-time support between students;
- **System design resources** that facilitate the visualization of the design of the systems developed: (i) ***Astah UML*** - tool used to model use cases specified and presented in the iterations; (ii) ***Balsamiq*** - tool used to create prototypes of systems; and (iii) ***Office 365 education*** - tool used to prepare presentations of the results of each iteration;
- **System development resources** that facilitate system development and ensure all teams have access to all projects: (i) ***GitHub*** - SECO platform used so the teams could construct their projects in the same environment, thus enabling students to contribute to all the projects from anywhere; and (ii) ***Other resources*** - the teams were free to use tools, support materials, and technologies of their choice in systems design and development such as ***Visual Studio***, ***Stack Overflow***, ***Vue.js*** and ***React***.

4 Evaluation Method

We conducted an observational study to gain insights into the influence relationships within the systems design and development discipline in a remote learning context, following the application of PA4SDD-SECO. An observational study involves social interaction between the researcher and the subjects, with data collected systematically throughout the research [20, 58, 72]. This study adhered to the recommendations by Seaman [61], who suggests conducting observations during meetings and utilizing emails to gather information that might otherwise go unnoticed, allowing for relevant notes to be taken for research purposes. Thus, we observed how

the characteristics of an influencer in SECO (Table 1) are reflected in the implementation of PA4SDD-SECO, as described in Section 3.

4.1 Real Case

The case was observed in a systems design and development course within the Information Systems program at a university, a higher education institution. Four researchers conducted the study, accompanying the students throughout their activities during the course. Typically, the course is offered face-to-face; however, it was delivered remotely due to the COVID-19 pandemic. It is important to highlight that the entire observational study took place during the application of PA4SDD-SECO, as described in Section 3. A professor with over 15 years of experience teaching Software Engineering reviewed and validated all study decisions. We organized the observed case as follows:

- We created the tasks and previously defined their deadlines in Moodle. In the students' panel, we included information about the course presentation, good practices for remote learning, links to synchronous remote classes, and support documents for the system's design and development;
- The course's 29 students (S1 to S29) made up 7 teams (Team 1 to Team 7) of 4 to 5 members of their choice. Thus, we defined two groups (A and B) for deliveries in alternate weeks to have space for discussion and not overload the synchronous remote classes (2 hours duration). Group A had 4 teams, and Group B had 3. The teams communicated spontaneously using the computational support described in Section 3.3;
- The first two classes were exploratory, with the students individually delivering a summary. The objective was to analyze the "teams" levels on systems analysis and design and system development processes besides commenting on the artifacts to be delivered during the discipline;
- Starting from the third class, the teams initiated the delivery of their project documentation. In this stage, they used tools such as Astah UML and Balsamiq. Each team member uploaded the document, with the respective video presenting the prototype in Moodle;
- In the following classes, the students developed the systems with the support of the common technological platform (GitHub). All teams had access to the platform to create their repository and could also access the repositories of other teams. Students delivered documentation with corrections and videos demonstrating their implementations in these classes;
- The teaching assistants and professor provided feedback and resolved student questions synchronously and asynchronously.

4.2 Collecting and Analyzing Data

We collected the data from observations in the synchronous remote meetings held with the teams and the information made available through the computational support defined in Section 3.3. Next, we listened to the teams' verbal explanations and wrote **observation notes** describing the actions of the teams' so that they could be coded and understood by any researcher. The first and second authors collected all data from this phase of the observational study without the students' knowledge, ensuring no direct interference from them in the research. To analyze the data, we performed an open and axial coding approach inspired by the initial procedures of the Grounded Theory [8]. We used notes and comments the students left on the computational support as well as the observation notes document for coding.

The first author conducted the coding process, as he had all the accumulated knowledge from observing the students. In open coding, he assigned each excerpt from the observation notes **preliminary** and **focused codes** and derived them iteratively. After open coding, he used axial coding to group the codes into **categories** [8] related to the characteristics of influencers in SECO (Table 11). The other authors reviewed the codes that emerged to ensure reliability. To do so, we organized the observation notes and codes into a document to track the results in several iterative cycles with discussions between the authors. With this procedure, we tried to mitigate the bias of the researcher's opinion, providing more reliability to the data and results. Table 2 shows examples of the coding process for transcripts and the resulting codes and categories.

We created a questionnaire on Google Forms, consisting of objective and subjective questions divided into five parts, to analyze students' perceptions of the PA4SDD-SECO application. The first part (1) presented an introductory text outlining the questionnaire's academic objectives. In the second part (2), participants read and indicated their agreement or disagreement with the consent form before proceeding to the questions. The third part (3) collected participant demographic and background information. The fourth part (4) presented students with six scenarios related to the characteristics of influencers in SECO, as described in Table 3. Students were asked to rate their perception of these characteristics during the course. We excluded the characteristics "*status in the project*" and "*status/popularity in GitHub*" since no differentiated permissions or privileges existed among students within the same project team. Participants rated each question on a scale from 1 (least observed) to 5 (most observed), indicating how prominently they perceived each scenario throughout the course. The fifth part (5) included questions about the overall evaluation of the course. Based on the questionnaire results, we analyzed the effectiveness of the PA4SDD-SECO application. The questionnaire is publicly available in an open repository[1].

5 Results

The systems design and development course was offered over a period of three months. It comprised 29 students, 4 teaching assistants (2 undergraduate and 2 graduate), and 1 professor. The students were divided into seven teams, organized into two groups (A and B) that alternated weekly deliveries. The professor and teaching assistants monitored submitted documents on Moodle, synchronous presentations via Google Meet, and activities on the common technological platform (GitHub). This approach enabled a comprehensive understanding of students' learning progression and the challenges encountered in conducting a practical course remotely.

[1] https://doi.org/10.5281/zenodo.10855251

Table 2: Illustration of the coding process.

Observation notes: *"The developers on the team knew the front-end and back-end. This made it easier to achieve a good result when delivering artifacts related to part of the project. Sometimes, developers S20 and S23 stayed up all night to project and execute on time."*		
Preliminary code	**Focused code**	**Category**
Developers knew the front-end and back-end	Participation with code	Technical
S20 and S23 stayed up all night to project and execute	Long-time interaction with the project	Social

Table 3: Scenarios and related characteristics.

ID	Scenario	Characteristics
1	*"One or more members are the team's source of expertise and are called upon to answer questions."*	Source of learning
2	*"A developer actively participates by contributing with substantial code to the project."*	Participation with code
3	*"The contributions of one member of the team were of high value for the project to achieve its objectives."*	Content value
4	*"A team member devoted much time to the project, whether it's documentation, development, or testing."*	Long interaction time with the project on GitHub
5	*"The comments, suggestions, and/or criticisms of a team member were important for the project development."*	Participation with comment
6	*"The good relationship among team members contributed to the project's success."*	Closeness to the GitHub project owner

Additionally, we analyzed the influence relationships among students throughout the course. Our results are presented following the educational strategy of PA4SDD-SECO outlined in Section 3.2.

5.1 Presentation of Syllabus and Project Definition

The professor presented the discipline at this stage. Next, the students asked questions about the system to be developed and the technologies and tools that could be used. After the explanations, the students were given 15 days to define the teams and an initial project proposal. Regarding team formation, some students had difficulties forming teams since they were geographically distant and did not know other students in the discipline. Hence, the synchronous remote classes and computational support helped improve interaction among them.

5.2 Review on Systems Design

At this stage, students became better acquainted with the professor and teaching assistants. We observed that establishing a direct communication channel between students, the professor, and teaching assistants, through the computational support defined in PA4SDD-SECO, was essential for project development. Additionally, students prepared summaries of the content to be discussed before synchronous remote classes, aiding in content review. Once teams were formed, students presented their initial project ideas. Teaching assistants also offered suggestions to help teams struggling to define their topics and held remote meetings to discuss project progress.

5.3 Presentation of Computational Support

The professor and teaching assistants presented the common technological platform (GitHub) as the system development environment. In addition, they also presented the system design resources (Astah and Balsamiq). The professor highlighted that students could choose the system development resources. We realized the teams did not use all the available resources in the GitHub. We observed that some students did not have experience using GitHub, making them use only the basics, the code repository, and the central part of the project. We also observed that in most teams, only one or two students defined the system development resources, such as the programming language and development frameworks.

5.4 System Development and Delivery

At this stage of the educational strategy of PA4SDD-SECO, we began to observe the influence relationships among students (who assumed the role of developers during the course) by identifying characteristics of influencers in SECO, as detailed next.

5.4.1 First Iteration. In the first iteration, teams delivered the initial project documentation, including requirements, business rules, system modeling, and prototypes. Difficulties emerged mainly in requirements elicitation and UML class diagram modeling. As coding had not yet started, the common technological platform (GitHub) was not used. We also observed the emergence of students acting as "product owners", concentrating knowledge of business rules and influencing team decisions by answering questions and commenting on peers' work, as shown in Table 4.

Table 4: The first iteration observations.

Characteristic	Observation
Participation with comments	S8, S12, S16, and S17 were responsible for commenting on their presentations and answering questions regarding their teams' projects. S5, S10, and S19 were responsible for commenting on their presentations and answering questions regarding their teams' projects.

5.4.2 Second Iteration. GitHub played an important role in system development from this iteration. Table 5 shows the relationships of influence between students identified at this stage. Building on feedback from the previous iteration, in which one team received positive comments for using a use case prioritization schedule, other teams adopted similar schedules. As S12 reported, "*we decided to use the schedule as we thought it was a promising idea from the other team*". This illustrates how practices spread across teams and how feedback influenced co-evolving projects in the SECO environment.

We also identified possible influencers with technical characteristics. Some teams adopted pair programming to balance knowledge between novice members and those with more experience. As S6 explained, "*it's preferable to sacrifice some initial time aligning knowledge rather than stopping several times during the project to explain*". During the presentations, some students responded promptly to technical questions. We noticed that some of these potential influencers contributed a lot of code to their projects.

Table 5: The second iteration observations.

Characteristic	Observation
Content value and source of learning	S2 and S22 were technical references in their teams, guiding the requirements prioritization.
Source of learning	S6 used pair programming to exchange experiences with novice students in systems design and development.
Participation with code	S2, S14, S20, and S23 were responsible for the many commits in their projects. S3, S11, and S29 were responsible for the many commits in their projects.

5.4.3 Third Iteration. Some teams reported difficulties in systems design and development and the need to change or remove use cases in this iteration. Collaboration extended beyond team boundaries, and one student reported calling a peer from another team to resolve a technical issue, exemplifying the SECO characteristic of interaction between internal and external actors (Table 6). When problems were not solved through peer support, students turned to external forums such as Stack Overflow, and two teams requested meetings with teaching assistants outside the synchronous classes. These meetings also contributed to observations related to the characteristics of influencers in SECO.

Table 6: The third iteration observations.

Characteristic	Observation
Source of learning	S14 mentioned that prior knowledge about some integrations made their use easier. S13 was the source of learning for his/her team members, and because of his/her previous experience with a specific programming language, members of other teams also asked him for help.

We also observed greater immersion in systems design and development. Despite clear role divisions, teams sought to share knowledge internally, with more experienced members mentoring novices. Students explicitly mentioned this exchange of experience in at least two teams. This aligns with Blincoe et al. [6], who state that developers tend to follow more experienced peers to learn from their code and practices.

5.4.4 Fourth Iteration. With final delivery approaching, some students reported the need to work for a long-time to complete the use cases. On the common technological platform, we observed that not all team members contributed code to their projects, as shown in Table 7. Typically, one or two students made most of the contributions. The students validated this observation by reporting a division of roles. According to them, the most complex parts of the project were aimed at the most experienced students in systems design and development. Some teams faced difficulties integrating their projects with external systems (e.g., authentication via social networking sites, use of maps, email systems etc.). These difficulties resulted in requirements changes in the final iterations. As some teams opted for similar technologies, there was a case where one team turned to a student from another team to resolve an integration issue. Other students reported sharing existing libraries and APIs so that different groups could reuse them, exemplifying the SECO characteristic of shared core assets.

Table 7: The fourth iteration observations.

Characteristic	Observation
Source of learning	Two teams reported that S2 and S11 were the main sources of learning for their respective team members due to their previous experience.
Long-time interaction with the project	Team 3 mentioned they had to spend nights fixing bugs and completing the project.
Participation with code	S6 and S29 made the most significant code contributions to their respective projects.
Participation with comments	S16 opened the issues on GitHub and encouraged discussions in his/her project.

Only one team used the GitHub issues area to report bugs or discuss new features. Although this feature was available, most students preferred to communicate through instant messaging applications, specifically WhatsApp, since they considered it more direct and practical. Some students also reported not using all the resources available on the platforms (e.g., GitHub issues) due to unfamiliarity or lack of prior experience.

5.4.5 Fifth Iteration. In the fifth and last iteration, all teams delivered projects with the minimum implemented requirements. One of the teams highlighted that an experienced student had chosen the technologies used in the project, which allowed him to complete the project promptly, as shown in Table 8. Some teams faced problems with time and difficulty for everyone to meet simultaneously. This difficulty may be due to remote teaching, where students face connection problems and available hours.

Table 8: The fifth Iteration observations.

Characteristic	Observation
Source of learning, and content value	The students pointed out that S14 was responsible for defining all the technologies used in the project.

Amidst the challenges faced in the COVID-19 pandemic, students reported that they often had to work on the project until

dawn, for several hours, so as not to compromise or delay deliveries. These data corroborate the study by Bozkurt and Sharma [7], which highlights that many students face difficulties in using their available time for studies remotely being necessary to create a habit for this type of teaching, in which the student themself knows how to organize their time. Another difficulty was the lack of financial conditions to acquire a computer and Internet, making it difficult to have quality and equal teaching and learning.

5.5 Final Assessment of the Projects

The evaluation of the projects was cumulative and related to the delivery of each team in the synchronous remote classes. Each delivery had a score of 100 points, corresponding to 80% of the grades. The remaining 20% were related to summaries delivered individually at the beginning of the course. Notes were made available to students via Moodle after each delivery in synchronous remote classes. The evaluation was carried out according to the quality of the projects delivered. There were no grades below average (50 points), and post-delivery comments aimed to help students not make the same mistakes. We emphasize that although the sum of the assessment only formally appears at the end of the process, intermediate estimates were present from the beginning of the subject, contributing to the high approval rate (100%). At the end, 25 of the 29 students obtained the maximum grade in their projects.

5.6 Analysis of the Pedagogical Architecture

We applied a questionnaire to analyze PA4SDD-SECO and students' perceptions of influence relationships. At the end, 21 of 29 students completed the questionnaire. Most participants (17 of 21) had prior experience with system development, and 20 of the 21 students reported having experience with GitHub. Regarding programming, ten reported high affinity, eight medium, and three low.

Regarding influence, the "*closeness to the GitHub project owner*" was the characteristic most noted by students. Those who already knew each other collaborated more easily, while others struggled to form teams due to lack of prior contact and the absence of in-person meetings during synchronous classes. This characteristic was also observed throughout the discipline. Figure 2 presents students' perceptions of influencer characteristics in their projects.

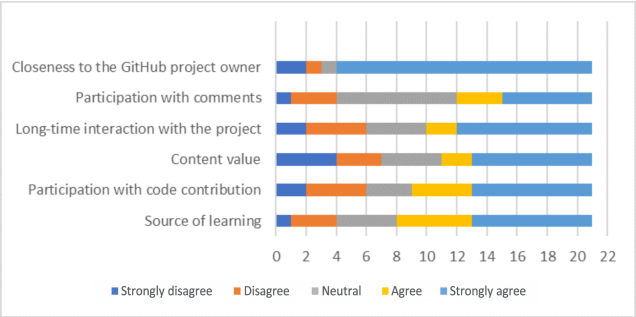

Figure 2: Characteristics of influencers perceived by students.

Regarding the educational strategy of PA4SDD-SECO, students highlighted the organization of deliveries in short iterations as a positive aspect. They also valued the flexibility and practical approach of the architecture. One respondent suggested incorporating pair programming to enhance learning across teams. We emphasized that students were free to choose how to develop their projects. Students also recommended providing more complementary materials and assigning a teaching assistant to each team. Table 9 presents the students' comments on the educational strategy.

Table 9: Comments on the educational strategy of PA4SDD-SECO.

ID.	Comment
S1	"It would be interesting to have a specific instructor for each team. Someone to check deliveries and talk directly to the teams."
S3	"I missed a more comprehensive set of materials that addressed subjects related to the discipline."
S7	"Encouraging pair programming among teams would further enable the knowledge exchange. The project enabled me to exchange knowledge, but I wished to have learned more about the code from other colleagues."
S10	"I liked the weekly delivery method. The two weeks between each delivery were essential for the good development of the projects."
S18	"The approach adds much more than tests and exercise lists. We learned a lot more and did not forget the concepts because we could apply the theory to our daily problems through the method adopted in the subject."

Regarding the computational support of PA4SDD-SECO, students considered direct contact with the professor and teaching assistants essential for the effective functioning of the course activities. Additionally, they suggested the adoption of other tools. Table 10 presents the students' comments.

Table 10: Comments on the computational support of PA4SDD-SECO.

ID.	Comment
S13	"I was pretty afraid of this discipline before starting. However, the organization through activities with deliveries planned and explained since the beginning of the discipline and open communication with the professor and teaching assistants made the discipline easier."
S16	"Using Google Classroom can lead to more fluent communication, but email was enough."

Thus, we noticed that the cooperation among the professor, students, and teaching assistants was a differential and essential communication for this exchange of knowledge and experiences. Another noteworthy point is the positive role of PA4SDD-SECO, which presented itself as promising support in developing systems, enabling the development of projects cooperatively and remotely.

6 Discussion

Our main results show that PA4SDD-SECO, based on SECO characteristics, can effectively support systems design and development in remote learning by fostering cooperation among students within and across teams. Furthermore, we observed multiple influence relationships between students through the identification of influencer characteristics in SECO (Table 11). This section discusses pedagogical approaches to systems design and development, as well as influence relationships among developers in SECO.

6.1 Pedagogical Approaches to Systems Design and Development

PA4SDD-SECO enables teams to collaboratively solve real-world problems, offering an authentic hands-on experience in systems design and development within a remote learning context. Students are also exposed to software quality aspects, such as decision traceability, consistent use of design artifacts, and progressive refinement of deliverables. Prior studies highlight the value of pedagogical approaches that approximate industry practice and provide enriching experiences in systems design and development [1, 3, 25, 27, 33, 48]. PA4SDD-SECO integrates pedagogical aspects with computational support to foster innovative educational proposals and is grounded in SECO principles. According to Damian et al. [17], SECO is becoming the predominant mode of software development.

PA4SDD-SECO encourages student autonomy by allowing them to define and adapt their system design and development strategies. One of its advantages is applicability to remote learning, overcoming geographical barriers and enabling practical cooperation. This flexibility is particularly relevant today, where remote education has become necessary [23, 46]. Applying PA4SDD-SECO showed that teamwork facilitates peer learning in remote settings. When teams include students with diverse backgrounds and experiences, they can cooperate and influence each other, benefiting the remote learning process. According to Porras et al. [53], adopting pedagogical approaches that promote team building during remote learning is crucial for computing courses. The authors also highlight that remote learning hinders the development of strong, lasting relationships between students and teachers, underscoring the importance of pedagogical strategies that foster synchronous and asynchronous team interactions during project [53].

PA4SDD-SECO also stimulates student protagonism by encouraging autonomy and responsibility for learning collaboratively through projects. This approach makes classes more dynamic and helps overcome geographical barriers, enabling cooperation even at a distance. According to Moumoutzis et al. [47], pedagogical strategies that guide students in managing projects enable effective cooperation in developing assigned software products, both in remote and classroom settings. Moreover, these strategies facilitate knowledge acquisition through interaction during project development [73] and encourage students to step outside their comfort zones [67]. Consequently, a student's performance depends not only on their individual efforts but also on the contributions of their teammates toward achieving project goals.

6.2 Relationship of Influence Between Developers in SECO

In the first deliveries of the system, we noticed a student influenced other students on his/her team and other teams through **"*participation with comments*"** during synchronous remote classes or on the common technological platform. According to Condina et al. [12], participation with comments is important in SECO-based scenarios. It can impact the project's direction and lead to decision making and improvements in design and code.

We also observed that the comments of the professor or teaching assistants also influenced the teams' decisions. For example, some teams changed their development framework due to feedback from teaching assistants or other students. By sharing their thoughts and discussing different perspectives, there was an exchange of experiences, allowing each person to contribute with their opinion. Cook et al. [13], Rocha et al. [57] and Kok et al. [37] state that the exchange of feedback between students is an essential activity for learning in computing courses, especially in open-ended and interaction-oriented domains, as in the PA proposed in this paper. Hamer et al. [26] and Stegeman et al. [65] corroborate this statement by describing that receiving feedback from other people helps receptors develop self-evaluation skills and learn on how to use it to make decisions during systems design and development.

We asked students how they organized the projects during systems design and development discipline. The students reported that more experienced students in systems design and development suggested ways to manage and execute the projects (**"*source of learning*"** and **"*content value*"**). Our results corroborate the study performed by Farias et al. [21]. The authors informed that experienced developers helped novice developers in order to help them overcome obstacles during the system's design and development.

We also identified that some potential influencers (more experienced students in systems design and development) were also contributing with code to their projects (**"*participation with code*"**) based on the common technological platform (GitHub). According to Cochez et al. [11], using GitHub as such a platform has several potential benefits for the professor. The professor can monitor students' progress, stimulate collaboration between students, track students' progress and understand how projects have evolved, and provide feedback efficiently. On the other hand, Suciu et al. [67] state that experience with these specific tools or platforms for project management and versioning supports the practical skills students will need in the job market, as they are the typical instruments used in real projects in software companies. In addition, Kalliamvakou et al. [34] describe how a transparent, open development environment fosters cooperation among software developers by supporting social coding platforms such as GitHub. This shifts communication and coordination from the project level to the network level of the SECO, as exemplified by PA4SDD-SECO. At the network level, actors define strategies for their immediate buyers and suppliers—for instance, by organizing meetings and maintaining communication and collaboration with other external stakeholders [30, 66].

We also asked the students if their previous physical proximity with other colleagues had helped them develop their activities. The students mentioned that knowing some of their colleagues beforehand helped them to form teams and establish a good relationship among team members, which in turn contributed to the delivery of the system (**"*closeness to the GitHub project owner*"**). This result confirms the findings of Silva et al. [62]. According to the authors, pedagogical approaches that promote teamwork with members they already know foster the exchange of knowledge.

Based on the observational study and students' responses to the questionnaire, we identified six out of eight characteristics of influencers in SECO (Table 1). We classified these characteristics according to theory of social influence [35], as shown in Table 11.

We noticed that the characteristic "***closeness to the GitHub project owner***" is linked to the "*identification*" social influence type. "*Identification*" is associated with accepting the influence of someone who is admired [35]. We observed that some students

Table 11: Characteristics versus social influence types by [35].

Characteristics	Social Influence Type
Source of learning	Compliance
Participation with code	Internalization
Content value	Internalization
Long-time interaction with the project	Internalization
Participation with comments	Internalization
Closeness to the GitHub project owner	Identification

already recognized and admired the know-how of these students since they already knew some members of their team.

The influencers' characteristics, "***participation with code***", "***content value***", and "***long-time interaction with the project***" were associated with the "*internalization*" social influence type. "*Internalization*" occurs when developers accept the influence due to the intrinsically rewarding nature of the influencer's content [35]. In our context, we observed that the novice students (with less experience in systems design and development) could not perform technical tasks on GitHub similarly to the more experienced students in systems design and development, even though there was an exchange of experiences during the iterations. However, the more experienced students exerted influence by helping novices resolve bugs, conduct tests, and review code.

The social influence type "*compliance*" is linked to "***source of learning***". "*Compliance*" occurs when individuals tend to act according to the expectations of the team to which they belong to achieve a certain goal [35]. In the case of novice students, we observed that they consensually accepted the influence of experienced students in systems design and development to get the system delivered on time. Therefore, the social influence type "*compliance*" can directly be related to "***source of learning***".

Studies focused on teaching and learning in computing courses such as computer science and information systems consider several social aspects, such as collaboration and cooperation [31, 38, 56]. However, other social aspects should be discussed in teaching and learning in computing courses, such as the relationship of influence between students. Professors need to be aware of the influencers in each project, either to avoid bad influences or to strengthen positive influences to support teaching and learning. A helpful strategy is to monitor discussions, considering not only students who contribute code but also those who provide ideas and comments on problems and code from other students.

7 Conclusion

This work proposed and applied a PA for systems design and development in remote learning based on SECO characteristics (PA4SDD-SECO). We applied PA4SDD-SECO in the discipline of systems design and development and observed the relationship of influence between the students who took on the role of developers during a discipline. Our results show that PA4SDD-SECO contributed to systems design and development in remote learning and improved student cooperation, regardless of geographic distance.

In the students' perception, PA4SDD-SECO was considered positive for supporting learning. Students emphasized that PA4SDD-SECO allowed cooperation between those in the same team and from different teams, even during remote learning. In students' perception, this collaborative environment favored software quality, encouraging reuse, feedback incorporation, and attention to artifact completeness. The knowledge sharing and cooperation between the students not only enriched the quality of the system but also helped to maintain a strict delivery schedule, which awakened an accountability relationship among the students in the teams. Team cooperation was key in ensuring that all the projects were delivered on time. This cooperative learning environment demonstrates how synergy between students can be a powerful tool for collective success. Moreover, we identified the relationship of influence between students. We observed that the knowledge exchange between novice and more experienced students contributed to the success of the projects. We also noticed that feedback directed to a team can influence other teams in the SECO context.

This work is limited to the analysis of classes within a one-course period. So, we did not seek to generalize the results but generate inputs to be considered in further academic experiences in other contexts. Furthermore, we must emphasize that the collected data refers to comments and perceptions of students participating in the course as well as researchers' reflections. Kitchenham et al. [36] state that a qualitative study addresses research questions related to the beliefs, experiences, attitudes, and opinions of human beings, either as individuals or in groups. Thus, qualitative studies aim to identify as many different viewpoints as possible rather than the most commonly expressed viewpoint. They also offer richer explanations and new areas for future study. We tried to address such a limitation by sharing data analysis and exchanging experiences with two more experienced researchers in the SECO context and in the course teaching (15 years).

As future work, we intend (i) to extend the study to other regions and/or subjects to verify whether the application of PA4SDD-SECO contributes to the teaching and learning process in other computing courses; (ii) to employ additional research methods (e.g., field studies) to compare influence relationships between novice and experienced developers in educational versus industrial contexts; (iii) to conduct further studies investigating whether a novice developer who is influenced can eventually become an influencer; and (iv) to employ objective metrics (e.g., collaboration networks on GitHub and quality of models/artifacts) to broaden the analysis of results.

ARTIFACT AVAILABILITY

The raw data and all the steps necessary to reproduce the study are detailed in the supplementary material located at https://doi.org/10.5281/zenodo.10855251.

ACKNOWLEDGEMENTS

The authors thank Coordination for the Improvement of Higher Education Personnel – Brazil (CAPES) – Financing Code 001, CNPq (Grants 316510/2023-8 and 311533/2025-6), FAPERJ (Grant E-26/204.404/2024), UNIRIO and UFRA for their support.

REFERENCES

[1] Joni K Adkins and Cindy Tu. 2021. Online Teaching Effectiveness: A Case Study of Online 4-Week Classes in a Graduate Information Systems Program. *Information Systems Education Journal* 19, 3 (2021), 31–37.

[2] Efthimia Aivaloglou and Anna van der Meulen. 2021. An Empirical Study of Students' Perceptions on the Setup and Grading of Group Programming Assignments. *ACM Transactions on Computing Education (TOCE)* 21, 3 (2021), 1–22. https://doi.org/10.1145/3440994

[3] Jacqueline Mayumi Akazaki, Bruna Kin Slodkowski, Leticia Rocha Machado, Kennya Ferreira Silva Miranda, Tássia Priscila Fagundes Grande, and Patricia Alejandra Behar. 2022. Pedagogical Strategies Based on Socio-affective Scenarios: An Outlook Based on Personalized Teaching in a Virtual Learning Environment. *Informatics in Education* 21, 4 (2022), 571–588. https://doi.org/10.15388/infedu.2022.23

[4] Manal M Alhammad and Ana M Moreno. 2018. Gamification in software engineering education: A systematic mapping. *Journal of Systems and Software* 141 (2018), 131–150. https://doi.org/10.1016/j.jss.2018.03.065

[5] Estela ML Aquino, Ismael Henrique Silveira, Julia Moreira Pescarini, Rosana Aquino, Jaime Almeida de Souza-Filho, Aline dos Santos Rocha, Andrea Ferreira, Audêncio Victor, Camila Teixeira, Daiane Borges Machado, et al. 2020. Social distancing measures to control the COVID-19 pandemic: potential impacts and challenges in Brazil. *Ciencia & saude coletiva* 25 (2020), 2423–2446. https://doi.org/10.1590/1413-81232020256.1.10502020

[6] Kelly Blincoe, Jyoti Sheoran, Sean Goggins, Eva Petakovic, and Daniela Damian. 2016. Understanding the popular users: Following, affiliation influence and leadership on GitHub. *Information and Software Technology* 70 (2016), 30–39. https://doi.org/10.1016/j.infsof.2015.10.002

[7] Aras Bozkurt and Ramesh C Sharma. 2020. Emergency remote teaching in a time of global crisis due to CoronaVirus pandemic. *Asian Journal of Distance Education* 15, 1 (2020), i–vi. https://doi.org/10.5281/zenodo.3778083

[8] Kathy Charmaz. 2006. *Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis*. Sage Publications.

[9] Steve Chenoweth and Panagiotis K. Linos. 2023. Teaching Machine Learning as Part of Agile Software Engineering. *IEEE Transactions on Education* (2023), 1–10. https://doi.org/10.1109/TE.2023.3337343

[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] Michael Cochez, Ville Isomöttönen, Ville Tirronen, and Jonne Itkonen. 2013. How do computer science students use distributed version control systems?. In *9th International Conference ICTERI 2013: Information and Communication Technologies in Education, Research, and Industrial Applications, Kherson, Ukraine, June 19-22, 2013, Revised Selected Papers 9*. 210–228. https://doi.org/10.1007/978-3-319-03998-5_11

[12] Vinicius Condina, Paulo Malcher, Victor Farias, Rodrigo Santos, Awdren Fontão, Igor Wiese, and Davi Viana. 2020. An Exploratory Study on Developers Opinions about Influence in Open Source Software Ecosystems. In *34th Brazilian Symposium on Software Engineering*. 137–146. https://doi.org/10.1145/3422392.3422404

[13] Amy Cook, Jessica Hammer, Salma Elsayed-Ali, and Steven Dow. 2019. How Guiding Questions Facilitate Feedback Exchange in Project-Based Learning. In *2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300368

[14] Lucinéia Barbosa da Costa, Orivaldo de Lira Tavares, Crediné Silva de Menezes, and Rosane Aragón. 2012. Pedagogical architectures and web resources in the teaching-learning of programming. In *Frontiers in Education Conference–FIE*.

[15] Emanuel F. Coutinho, Italo Santos, Leonardo O. Moreira, and Carla I. M. Bezerra. 2019. A Report on the Teaching of Software Ecosystems in Software Engineering Discipline. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*. Association for Computing Machinery, 130–139. https://doi.org/10.1145/3350768.3351302

[16] Emanuel Ferreira Coutinho, Davi Viana, and Rodrigo Pereira Dos Santos. 2017. An Exploratory Study on the Need for Modeling Software Ecosystems: The Case of SOLAR SECO. In *9th International Workshop on Modelling in Software Engineering (MiSE)*. 47–53. https://doi.org/10.1109/MiSE.2017.3

[17] Daniela Damian, Johan Linåker, David Johnson, Tony Clear, and Kelly Blincoe. 2021. Challenges and Strategies for Managing Requirements Selection in Software Ecosystems. *IEEE Software* 38, 6 (2021), 76–87. https://doi.org/10.1109/MS.2021.3105044

[18] Thalia Santos de Santana, Taciana Novo Kudo, Davi Viana, and Renato F. Bulcão-Neto. 2025. Professors' Perspective on a Pedagogical Architecture to Requirements Engineering Education: A Qualitative Study. *Journal of Software Engineering Research and Development* 13 (2025), 10. https://doi.org/10.5753/jserd.2025.4567

[19] Rodrigo Duran, Silvia Amélia Bim, Itana Gimenes, Leila Ribeiro, and Ronaldo Celso Messias Correia. 2023. Potential Factors for Retention and Intent to Dropout in Brazilian Computing Programs. *ACM Trans. Comput. Educ.* 23, 3, Article 36 (sep 2023), 33 pages. https://doi.org/10.1145/3607537

[20] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. 2008. Selecting empirical methods for software engineering research. *Guide to Advanced Empirical Software Engineering* (2008), 285–311. https://doi.org/10.1007/978-1-84800-044-5_11

[21] Victor Farias, Igor Wiese, and Rodrigo Santos. 2019. What characterizes an influencer in software ecosystems? *IEEE Software* 36, 1 (2019), 42–47. https://doi.org/10.1109/MS.2018.2874325

[22] Fabrício Wickey Silva Garcia, Sandro Ronaldo Bezerra Oliveira, and Elielton da Costa Carvalho. 2022. A Second Experimental Study the Application of a Teaching Plan for the Algorithms Subject in an Undergraduate Course in Computing using Active Methodologies. *Informatics in Education* 22, 2 (2022), 233–255. https://doi.org/10.15388/infedu.2023.12

[23] Miguel Garcia, Jose Quiroga, and Francisco Ortin. 2021. An Infrastructure to Deliver Synchronous Remote Programming Labs. *IEEE Transactions on Learning Technologies* 14, 2 (2021), 161–172. https://doi.org/10.1109/TLT.2021.3063298

[24] Cayley Guimarães, Diego R. Antunes, Laura S. García, Letícia M. Peres, and Sueli Fernandes. 2013. Pedagogical Architecture – Internet Artifacts for Bilingualism of the Deaf (Sign Language/Portuguese). In *2013 46th Hawaii International Conference on System Sciences*. 40–49. https://doi.org/10.1109/HICSS.2013.445

[25] Olival Gusmão, Marcus de Melo Braga, and Victor Diogho Heuer de Carvalho. 2021. Applying Strategic Planning in a Distance Undergraduate Course in Information Systems: A Case Study. In *Trends and Applications in Information Systems and Technologies*. 42–51. https://doi.org/10.1007/978-3-030-72660-7_5

[26] John Hamer, Helen Purchase, Andrew Luxton-Reilly, and Paul Denny. 2015. A comparison of peer and tutor feedback. *Assessment & Evaluation in Higher Education* 40, 1 (2015), 151–164. https://doi.org/10.1080/02602938.2014.893418

[27] Markus Helfert. 2011. Characteristics of information systems and business informatics study programs. *Informatics in Education-An International Journal* 10, 1 (2011), 13–36. https://doi.org/10.15388/infedu.2011.02

[28] Maria-Blanca Ibanez, Angela Di-Serio, and Carlos Delgado-Kloos. 2014. Gamification for engaging computer science students in learning activities: A case study. *IEEE Transactions on Learning Technologies* 7, 3 (2014), 291–301. https://doi.org/10.1109/TLT.2014.2329293

[29] Mohammed Mansur Ibrahim and Muesser Nat. 2019. Blended learning motivation model for instructors in higher education institutions. *International Journal of Educational Technology in Higher Education* 16, 1 (2019), 1–21. https://doi.org/10.1186/s41239-019-0145-2

[30] Slinger Jansen, Anthony Finkelstein, and Sjaak Brinkkemper. 2009. A sense of community: A research agenda for software ecosystems. In *31st International Conference on Software Engineering - Companion Volume*. 187–190. https://doi.org/10.1109/ICSE-COMPANION.2009.5070978

[31] Ângelo Magno de Jesus Jesus and Ismar Frango Silveira. 2022. A Collaborative Learning Framework for Computational Thinking Development through Game Programming. *Informatics in Education* 21, 2 (2022), 253–281. https://doi.org/10.15388/infedu.2022.14

[32] Yuchao Jiang, Daniel Schlagwein, and Boualem Benatallah. 2018. A Review on Crowdsourcing for Education: State of the Art of Literature and Practice. *PACIS* (2018), 1–14. https://doi.org/pacis2018/180

[33] Juho Kahila, Henriikka Vartiainen, Matti Tedre, Eetu Arkko, Anssi Lin, Nicolas Pope, Ilkka Jormanainen, and Teemu Valtonen. 2024. Pedagogical framework for cultivating children's data agency and creative abilities in the age of AI. *Informatics in Education* 23, 2 (2024), 323–360. https://doi.org/10.15388/infedu.2024.15

[34] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. 2016. An in-depth study of the promises and perils of mining GitHub. *Empirical Software Engineering* 21, 5 (2016), 2035–2071. https://doi.org/10.1007/s10664-015-9393-5

[35] Herbert C Kelman. 1958. Compliance, identification, and internalization three processes of attitude change. *Journal of Conflict Resolution* 2, 1 (1958), 51–60. https://doi.org/10.1177/002200275800200106

[36] Barbara Ann Kitchenham, David Budgen, and Pearl Brereton. 2015. *Evidence-based software engineering and systematic reviews*. Vol. 4. CRC Press.

[37] Arjan JF Kok, Lex Bijlsma, Cornelis Huizing, Ruurd Kuiper, and Harrie Passier. 2024. Analysis and Evaluation of a Searchable Exercise Repository for Training Java Programming. *Informatics in Education* 23, 2 (2024), 361–383. https://doi.org/10.15388/infedu.2024.17

[38] Luis la Torre, Ruben Heradio, Carlos A Jara, Jose Sanchez, Sebastian Dormido, Fernando Torres, and Francisco A Candelas. 2013. Providing collaborative support to virtual and remote laboratories. *IEEE Transactions on Learning Technologies* 6, 4 (2013), 312–323. https://doi.org/10.1109/TLT.2013.20

[39] Elise Lavoué, Baptiste Monterrat, Michel Desmarais, and Sébastien George. 2018. Adaptive Gamification for Learning Environments. *IEEE Transactions on Learning Technologies* 12, 1 (2018), 16–28. https://doi.org/10.1109/TLT.2018.2823710

[40] Dagyeom Lee and Youngjun Lee. 2024. Productive failure-based programming course to develop computational thinking and creative Problem-Solving skills in a Korean elementary school. *Informatics in Education* 23, 2 (2024), 385–408. https://doi.org/10.15388/infedu.2024.14

[41] Daniela Lettner, Florian Angerer, Herbert Prähofer, and Paul Grünbacher. 2014. A Case Study on Software Ecosystem Characteristics in Industrial Automation Software. In *2014 International Conference on Software and System Process* (Nanjing,

China). Association for Computing Machinery, New York, NY, USA, 40—-49. https://doi.org/10.1145/2600821.2600826

[42] Jefferson Ribeiro Lima and Crediné Silva Menezes. 2025. Uma Arquitetura pedagógica para apoiar a aprendizagem de programação introdutória no ensino superior. *RENOTE* 23, 1 (2025), 218–229.

[43] Orivaldo Lira Tavares, Crediné Silva de Menezes, and Rosane Aragon de Nevado. 2012. Pedagogical architectures to support the process of teaching and learning of computer programming. In *2012 Frontiers in Education Conference*. 1–6. https://doi.org/10.1109/FIE.2012.6462427

[44] Mircea Lungu, Michele Lanza, Tudor Gîrba, and Romain Robbes. 2010. The small project observatory: Visualizing software ecosystems. *Science of Computer Programming* 75, 4 (2010), 264–275. https://doi.org/10.1016/j.scico.2009.09.004

[45] Konstantinos Manikas. 2016. Revisiting software ecosystems research: A longitudinal literature study. *Journal of Systems and Software* 117 (2016), 84–103. https://doi.org/10.1016/j.jss.2016.02.003

[46] Maria José Marcelino, Teresa Pessoa, Celeste Vieira, Tatiana Salvador, and António José Mendes. 2018. Learning computational thinking and scratch at distance. *Computers in Human Behavior* 80 (2018), 470–477. https://doi.org/10.1016/j.chb.2017.09.025

[47] Nektarios Moumoutzis, George Boukeas, Vassilis Vassilakis, Nikos Pappas, Chara Xanthaki, Ioannis Maragkoudakis, Antonios Deligiannakis, and Stavros Christodoulakis. 2018. Design, Implementation and Evaluation of a Computer Science Teacher Training Programme for Learning and Teaching of Python Inside and Outside School. In *Interactive Mobile Communication Technologies and Learning: 11th IMCL Conference*. 575–586. https://doi.org/10.1007/978-3-319-75175-7_56

[48] Vânia de Oliveira Neves, Silvana Morita Melo, Davi Viana, Rodrigo Pereira dos Santos, and Valdemar Vicente Graciano Neto. 2023. Challenges on the Brazilian Information Systems Education: The Professors' Perspective. *IEEE Transactions on Education* 66, 6 (2023), 531–542. https://doi.org/10.1109/TE.2023.3259335

[49] Daniel Olivares, Christopher Hundhausen, and Namrata Ray. 2021. Designing IDE Interventions to Promote Social Interaction and Improved Programming Outcomes in Early Computing Courses. *ACM Trans. Comput. Educ.* 22, 1, Article 2 (oct 2021), 29 pages. https://doi.org/10.1145/3453165

[50] Maximiliano Paredes-Velasco, Mónica Arnal-Palacián, Jaime Urquiza-Fuentes, and Mercedes Martín-Lope. 2023. Improving Soft Skills Through an Interdisciplinary Approach in a Realistic Context Between Education and CS Students in an HCI Course. *IEEE Transactions on Education* 66, 6 (2023), 579–590. https://doi.org/10.1109/TE.2023.3269691

[51] Oskar Pettersson and Didac Gil. 2010. On the Issue of Reusability and Adaptability in M-learning Systems. In *2010 6th IEEE International Conference on Wireless, Mobile, and Ubiquitous Technologies in Education*. 161–165. https://doi.org/10.1109/WMUTE.2010.48

[52] Jean Piaget. 1972. Intellectual evolution from adolescence to adulthood. *Human Development* 15, 1 (1972), 1–12. https://doi.org/10.1159/000271225

[53] Jari Porras, Ari Happonen, and Jayden Khakurel. 2021. Experiences and Lessons Learned from Onsite and Remote Teamwork Based Courses in Software Engineering. In *2021 International Conference on Data and Software Engineering (ICoDSE)*. 1–9. https://doi.org/10.1109/ICoDSE53690.2021.9648490

[54] Filipe Portilho, João Amaral, Nicole Rodrigues, Cleon Pereira Junior, and Newarney Costa. 2024. Uma Arquitetura Pedagógica para o Ensino de Lógica de Programação: Lições Aprendidas a partir de um Projeto de Extensão. In *XXXII Workshop sobre Educação em Computação* (Brasília/DF). SBC, Porto Alegre, RS, Brasil, 329–340. https://doi.org/10.5753/wei.2024.3053

[55] David Brito Ramos, Ilmara Monteverde Martins Ramos, Alberto Castro, and Elaine Harada Teixeira de Oliveira. 2021. Collaborative Content Construction: A Pedagogical Architecture to support distance education. In *Workshop on Advanced Virtual Environments and Education*. 15–25. https://doi.org/10.5753/wave.2020.212070

[56] Evan F Risko, Tom Foulsham, Shane Dawson, and Alan Kingstone. 2012. The collaborative lecture annotation system: A new TOOL for distributed learning. *IEEE Transactions on Learning Technologies* 6, 1 (2012), 4–13. https://doi.org/10.1109/TLT.2012.15

[57] Hemilis Joyse Barbosa Rocha, Patrícia Cabral De Azevedo Restelli Tedesco, and Evandro De Barros Costa. 2022. On the use of feedback in learning computer programming by novices: a systematic literature mapping. *Informatics in Education* 22, 2 (2022), 209–232. https://doi.org/10.15388/infedu.2023.09

[58] Nyyti Saarimäki. 2020. Methodological Issues in Observational Studies. *SIGSOFT Softw. Eng. Notes* 44, 3 (oct 2020), 33–42. https://doi.org/10.1145/3356773.3356799

[59] Rodrigo Pereira Santos and Cláudia Werner. 2011. Treating business dimension in software ecosystems. In *International Conference on Management of Emergent Digital EcoSystems*. 197–201. https://doi.org/10.1145/2077489.2077526

[60] Marlene Scardamalia and Carl Bereiter. 2010. A brief history of knowledge building. *Canadian Journal of Learning and Technology/La revue canadienne de l'apprentissage et de la technologie* 36, 1 (2010). https://doi.org/10.21432/T2859M

[61] Carolyn B. Seaman. 1999. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering* 25, 4 (1999), 557–572. https://doi.org/10.1109/32.799955

[62] Leonardo Humberto Silva, Renata Xavier Castro, and Marice Costa Guimaraes. 2021. Supporting Real Demands in Software Engineering with a Four Steps Project-Based Learning Approach. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. 50–59. https://doi.org/10.1109/ICSE-SEET52601.2021.00014

[63] Lidia M. Silva, Lucas P. S. Dias, Jorge L. V. Barbosa, Sandro J. Rigo, Julio C. S. dos Anjos, Claudio F. R. Geyer, and Valderi R. Q. Leithardt. 2022. Learning analytics and collaborative groups of learners in distance education: a systematic mapping study. *Informatics in Education* 21, 1 (2022), 113–146. https://doi.org/10.15388/infedu.2022.05

[64] Ivan Srba and Maria Bielikova. 2015. Dynamic Group Formation as an Approach to Collaborative Learning Support. *IEEE Transactions on Learning Technologies* 8, 2 (2015), 173–186. https://doi.org/10.1109/TLT.2014.2373374

[65] Martijn Stegeman, Erik Barendsen, and Sjaak Smetsers. 2016. Designing a Rubric for Feedback on Code Quality in Programming Courses. In *16th Koli Calling International Conference on Computing Education Research*. 160—-164. https://doi.org/10.1145/2999541.2999555

[66] Daniel Ståhl, Kristian Sandahl, and Lena Buffoni. 2022. An Eco-System Approach to Project-Based Learning in Software Engineering Education. *IEEE Transactions on Education* 65, 4 (2022), 514–523. https://doi.org/10.1109/TE.2021.3137344

[67] Dan Mircea Suciu, Simona Motogna, and Arthur-Jozsef Molnar. 2023. Transitioning a project-based course between onsite and online. An experience report. *Journal of Systems and Software* 206 (2023), 1–10. https://doi.org/10.1016/j.jss.2023.111828

[68] Ville Tirronen and Ville Isomöttönen. 2011. Making teaching of programming learning-oriented and learner-directed. In *11th Koli Calling International Conference on Computing Education Research*. 60–65. https://doi.org/10.1145/2094131.2094143

[69] George Valença and Carina Alves. 2016. Understanding how power influences business and requirements decisions in software ecosystems. In *31st Annual ACM Symposium on Applied Computing*. 1258–1263. https://doi.org/10.1145/2851613.2851756

[70] George Valença, Nycolas Lacerda, Maria Eduarda Rebelo, Carina Alves, and Cleidson RB de Souza. 2019. On the benefits of corporate Hackathons for software ecosystems–a systematic mapping study. In *Product-Focused Software Process Improvement: 20th International Conference, PROFES 2019, Barcelona, Spain, November 27–29, 2019, Proceedings 20*. 367–382. https://doi.org/10.1007/978-3-030-35333-9_27

[71] Chris Wilcox and Albert Lionelle. 2018. Quantifying the benefits of prior programming experience in an introductory computer science course. In *49th ACM Technical Symposium on Computer Science Education*. 80–85. https://doi.org/10.1145/3159450.3159480

[72] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science & Business Media. https://doi.org/10.1007/978-3-642-29044-2

[73] Awad A Younis, Rajshekhar Sunderraman, Mike Metzler, and Anu G Bourgeois. 2021. Developing parallel programming and soft skills: A project based learning approach. *J. Parallel and Distrib. Comput.* 158 (2021), 151–163. https://doi.org/10.1016/j.jpdc.2021.07.015

[74] Lanqin Zheng, Miaolang Long, Bodong Chen, and Yunchao Fan. 2023. Promoting knowledge elaboration, socially shared regulation, and group performance in collaborative learning: an automated assessment and feedback approach based on knowledge graphs. *International Journal of Educational Technology in Higher Education* 20, 1 (2023), 46. https://doi.org/10.1186/s41239-023-00415-4