Do's and Don't s of Partnering with Industry to Educate Software Engineering Students: Recommendations Based on a Teaching Experience

Natalya Marjana Goelzer
School of Technology
Pontifical Catholic University of Rio Grande do Sul
Porto Alegre, RS, Brazil
natalya.goelzer@edu.pucrs.br

Pedro Portella Possamai
School of Technology
Pontifical Catholic University of Rio Grande do Sul
Porto Alegre, RS, Brazil
pedro.portella@edu.pucrs.br

Sabrina Marczak
School of Technology
Pontifical Catholic University of Rio Grande do Sul
Porto Alegre, RS, Brazil
sabrina.marczak@pucrs.br

ABSTRACT

This experience report describes a software development training program for undergraduate students, resulting from a long-standing collaboration between the Pontifical Catholic University of Rio Grande do Sul (PUCRS) and a renowned multinational in the technology industry, one of the largest retailers in the global IT market. The program aims to prepare students for professional software development before they formally join industry teams. Established over 25 years ago, the program currently trains around 60 students per year and has followed its current structure since 2016. This paper characterizes the university-industry partnership model and presents an overview of the program's four main phases: (1) participant selection, (2) technical training, (3) hands-on practice, and (4) industry internship. The first three phases take place within the university. The discussion highlights key challenges and successes identified throughout each stage, and analyzes how early exposure to industry-aligned practices helps bridge the gap between academic learning and real-world professional demands. The experience report also synthesizes lessons learned through a summary of Do's and Don't s for each phase, offering practical recommendations for similar initiatives.

KEYWORDS

Education, University-Industry Partnership, Software Engineering, Student Training, Recommendations

1 Introduction

The lack of alignment between academic curricula and the demands of the software industry has generated recurring concerns in the educational and professional environment. The fast evolution of tools, methodologies and practices in the sector makes it challenging for higher education universities to update their programs at the same speed, which results in gaps between the training offered and the profile demanded by the sector [9, 21, 22, 36]. This gap affects the integration of graduates into the job market and, consequently, the capacity of to train qualified professionals on a large scale [29].

The limitations are not restricted to technical training. Students often also lack the interpersonal skills essential for teamwork, problem-solving and effective communication [8]. Moreover, many interns enter the workforce without practical experience in real development processes, collaboration routines, or software engineering standards. This paper addresses that common challenge in software engineering education: the gap between what students typically learn and the skills expected by the industry. The search for complementary approaches to graduation has intensified, especially those that promote more effective integration between theory and practice. Training programs conducted in collaboration between universities and companies have shown promise in this scenario [1, 9, 38]. Initiatives that combine active learning environments and real projects have achieved good results in preparing students for the job market [1, 5, 18]. These programs expand training by offering practical experiences in authentic contexts, encouraging interaction, decision-making and the development of technical and interpersonal skills [11, 14].

This paper presents an experience report on the IT Program, an educational initiative resulting from a long-standing partnership between Pontifical Catholic University of Rio Grande do Sul (PUCRS) and a multinational technology company. Created over 25 years ago, the program complements the academic training of undergraduate IT students by preparing them to work on real-world projects, fostering both technical and behavioral skills aligned with industry needs. The program was specifically designed by the university and the company to bridge the gap between academic learning and industrial demands through internal training that integrates interns into real environments from day one. Unlike other initiatives that simulate professional contexts, the IT Program embeds participants under actual production constraints. While other programs, such as those discussed in [18], emphasize real projects and collaboration, our model stands out for the depth of integration: this is a production environment, not a lab or prototype. This setup accelerates the development of both technical and soft skills, as reflected in the program's high retention and hiring rates.

The program is organized in cohorts, each comprising approximately 20 interns. Its current structure, established in 2016, has been adopted across 21 completed cohorts. Designed to be adaptive and evolving, the structure is reviewed and updated after each edition through continuous collaboration between the university and the company, ensuring alignment with technological and strategic changes. This process of constant improvement keeps the program in line with changes in the sector. With a hiring rate of 96% and reports of accelerated career progression for participants within the organization, the initiative demonstrates its solidity and effectiveness in training professionals for the technology area. Unlike other initiatives that simulate workplace environments, this program places interns in real production settings from day one. This report

contributes by presenting a complete and replicable training model that includes academic design, industry integration, and shared governance. Additionally, it highlights how the industry benefits from adopting educational strategies, making the university an active contributor to talent development within the company.

The IT Program consists of four sequential stages: (1) participant selection, (2) technical training, (3) hands-on practice, and (4) company internship. The first three take place at the university, while the final stage is conducted at ORG's site, where interns integrate into professional teams. This experience report highlighting the partnership between university and industry in training students for the technology sector. The analysis focuses on the contributions of each partner throughout the different phases of the program, as well as on the lessons learned over the years, with the aim of providing input for other initiatives of collaboration between academia and industry. The paper is organized as follows: Section 2 presents the general context of the training program; Section 3 describes the teaching methodologies adopted; Section 4 details the structure and execution of the four program stages; Section 5 discusses the university-industry partnership and the distribution of responsibilities between the involved institutions; and Section 6 systematizes the main lessons learned.

2 Teaching Methodologies

Current approaches to software engineering education have increasingly adopted pedagogical models that place the student at the center of the learning process. These models are grounded in principles of active learning, which promote critical thinking, collaboration, and the practical application of knowledge. Active learning methodologies have proven effective in developing both technical and behavioral competencies essential for professional practice in the software industry [3, 27, 43]. In this context, students are encouraged to participate in learning activities that involve reasoning, creativity, decision-making, and teamwork. Instructors take on the role of facilitators, guiding the learning process and proposing real or simulated challenges that foster autonomy and reflective practice [2, 3, 32].

Two approaches widely used in this context are Problem-Based Learning (PBL) and Project-Based Learning (PjBL) [2, 5, 11, 31, 39, 40]. PBL is based on the analysis of complex problems to stimulate the search for solutions and the articulation between theory and practice. PjBL, on the other hand, focuses on the development of concrete projects, which require planning, collaboration and the integrated application of knowledge over several stages . These approaches are in line with the principles of constructivist learning, which understands knowledge as something that is actively constructed, based on the student's experiences and interactions with the environment and with others [12, 15, 17, 36]. And they are also related to the concept of situated learning, according to which knowledge is more meaningful when it is developed in contexts close to professional practice. From this perspective, participation in activities that simulate the real work environment contributes to the development of technical and social skills relevant to working in the software area [23, 25].

Another important pedagogical principle is collaborative learning, which emphasizes teamwork, shared responsibility, and collective problem-solving [4]. These practices are fundamental to the dynamics of software development and contribute to the acquisition of both technical skills and interpersonal competencies [10, 40]. Within this approach, pair programming stands out as an effective technique for collaborative knowledge building, as it promotes the exchange of ideas, increases engagement, enhances code quality, and reduces individual frustration [41, 42]. Additionally, the role of mentoring in the learning process has gained prominence in educational strategies aimed at developing complex skills. Mentoring involves modeling behavior, offering guidance, and providing timely feedback, contributing not only to the learner's technical development but also to their professional growth [7, 33]. It fosters a culture of support, trust, and mutual learning, which is particularly valued in computing education contexts [13, 26]. The teaching approaches presented in this section support the structure and implementation of the training program described in the next section.

Some initiatives have strengthened collaboration between industry and academia, including in-company graduate training [34], remote specialization programs [20], and project-based learning in undergraduate and graduate courses [6, 35]. These efforts emphasize real projects and industry integration, but often involve experienced professionals or simulate workplace contexts without formal employment. In contrast, the IT Program embeds early-career students directly into production environments from day one, under formal internships, combining academic and industry mentoring to develop both technical and behavioral skills. While [20, 34] address experienced professionals, and [6, 35] simulate real scenarios without employment ties, the IT Program integrates these dimensions in a structured and immersive way. It also incorporates behavioral development practices such as empathy guidelines [37], retrospectives [30], and coaching [24] as part of daily routines. Additionally, it addresses curricular alignment with industry demands, a recurring concern in the literature [19, 28], through a four-phase model with shared governance and defined quality standards.

3 IT Program

The IT Program results from a strategic collaboration between PUCRS and a multinational technology company, referred to in this paper as ORG. Established over 25 years ago, this partnership bridges academic education with the practical demands of software development, aiming to align the profile of young professionals (understood as undergraduate students who have not yet completed their degrees) with the operational needs of the company's development teams.

The IT Program prepares IT undergraduates to intern on ORG projects, fostering both technical and behavioral skills. Its current structure, in place since 2016, has supported 21 cohorts, each with around 20 participants, and runs two to three editions per year. To apply, students must be regularly enrolled in an undergraduate IT program (bachelor's or technologist), reside in the Porto Alegre region where PUCRS and ORG are located and have at least one year of availability to dedicate 30 hours per week to the program, with a minimum of four hours in the afternoon.

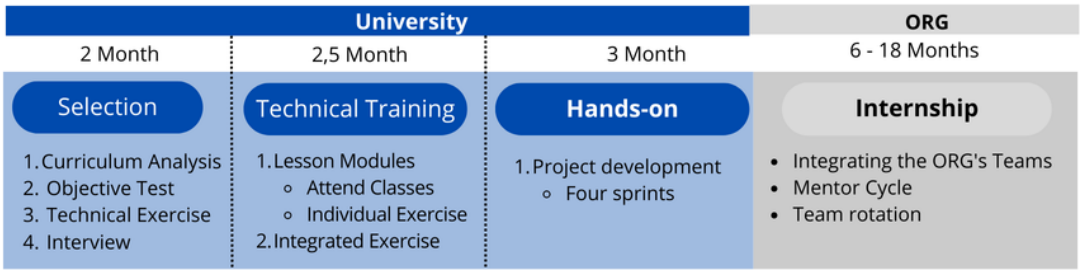

Figure 1: IT Program Structure

In this structure, PUCRS is responsible for designing and conducting pedagogical activities, ensuring alignment with ORG's expectations regarding skill development. ORG, in turn, identifies and continuously updates the set of competencies required by its teams, aiming to accelerate interns' learning and ensure the relevance of the educational content to current industry needs.

The training simulates a real work environment, emphasizing industry-standard practices, tools, processes, and quality standards. The collaboration between PUCRS and ORG promotes an integrated approach that combines academic knowledge with professional demands. The program targets the development of hard skills—such as programming, databases, industry relevant technologies, understanding software development stages, tool usage, and participation in agile ceremonies as well as soft skills like communication, teamwork, autonomy, and critical thinking. The following sections present the structure of program's design.

4 Partnership Model

The Program's partnership model is structured across three levels: executive, coordination, and operational, each with specific responsibilities. At the executive level, one manager from ORG and one from PUCRS are responsible for the program's design and vision. This leadership ensures alignment with industry needs, sets long-term goals, approves resource allocation, evaluates results, renews cooperation agreements, and maintains institutional commitment.

The coordination level includes representatives from both institutions who plan and guide the program's activities. This team connects the executive and operational levels, implements defined strategies, updates the curriculum based on past editions, and ensures integration of theory and practice with a focus on active learning. It also monitors learning indicators and participants' progress, ensuring pedagogical and technical goals are met.

At the operational level, professionals execute the planned activities. PUCRS operates through the Software Engineering Research Center (CEPES), managing contracts, schedules, evaluations, and team communication. Instructors, assistants, and the Scrum Master conduct the technical training and hands-on sessions, supervise practical work, provide feedback, and support learning. ORG's Product Owners(PO) contribute real project demands and technical guidance during the practical phase.

The program's structure was co-designed and is continually refined through this three-level model. Responsibilities are shared throughout all phases, ensuring that both institutions actively contribute to planning, execution, and evaluation. ORG aims to accelerate the integration of interns into development teams by promoting technical and interpersonal skill development, familiarizing them with tools and practices, and enabling structured feedback from academic mentors. PUCRS is responsible for delivering educational content, mentoring, and ensuring alignment with ORG's evolving needs. This multi-level structure helps distribute responsibilities according to each institution's strengths and maintain goal alignment. With clearly defined roles, the collaboration advances smoothly and enables continuous refinement.

5 Program Structure

The IT Program is structured in four sequential stages: (1) selection of participants, (2) technical training, (3) hands-on practice and (4) internship experience at the company (see Figure 1). The first three stages take place at the PUCRS's campus, while the last takes place at ORG's site, where participants experience the company's professional environment. The distribution of activities between the two institutions follows the established partnership model, respecting the decision-making levels and responsibilities assigned to each side. The organization of the stages is designed to provide a gradual learning path that combines theoretical training, practical application and professional immersion. Each phase of the program is presented in detail below.

5.1 Selection Process

IT Program participants are selected through a structured and collaborative selection process between the PUCRS and the ORG. This stage directly involves the partnership's three management levels, whose responsibilities are distributed throughout the recruitment process, ensuring consistency between the program's strategic objectives and the practical conduct of the activities.

At the beginning of each selection process, representatives from the executive level of PUCRS and ORG meet to define the strategic guidelines for the process. At these meetings, the managers determine the number of open positions, the profile of the candidates and the number of interviews the company will conduct. These decisions guide all the other stages of the selection process.

At the operational level, the university carries out the practical actions that make the selection process possible. The CEPES organizes the selection schedule, ensuring that deadlines are respected. The team prepares the public call for the selection process and disseminates it on institutional channels and social networks,

ensuring broad visibility and reach to the program's target audience. Instructors and assistant design the evaluation instruments, supervise the application of the in-person tests and evaluated the results of with attention to the pre-defined criteria.

The selection takes place in four elimination phases:

(1) **Curriculum Analysis:** The CEPES team checks the documentation sent in and assesses if the candidates meet the eligibility criteria previously established.
(2) **Objective test:** The objective test is administered in person by the university and consists of 30 multiple-choice questions divided into four content areas: programming logic, computer fundamentals, logical reasoning, and English. The application and supervision of the test are coordinated by instructors, assistants, and CEPES. Instructors and assistants are also responsible for create the tests and evaluating the results.
(3) **Technical Exercise:** This is a remote practical challenge in which candidates receive a system description containing functional requirements. They are expected to design and implement a functional and well-structured solution, using the programming languages and tools of their choice. The challenge is prepared by instructors and assistants, who also evaluate the submitted solutions based on multiple criteria, including code clarity and organization, adherence to requirements, creativity, and overall technical quality.
(4) **Interview:** The candidates with the best performance in the previous phases take part in interviews, conducted by representatives from University and ORG. The teams measure soft skills, such as communication, proactivity and teamwork, in order to complement the technical assessments and identify the most fitting profiles for the company's projects.

The approved students are officially hired as ORG interns and as part of the current edition of the program. According to Brazilian law, these interns can remain in this position for up to two years.

5.2 Technical Training

Technical Training (see Figure 2) is conducted with a focus on promote the knowledge needed for the hands-on. The planning of this phase involves the integrated action of different management levels. At the executive level, institutional representatives from the University and the ORG collaborate to define the content that will be covered in each edition of the program. These decisions are made based on the technical demands of the projects planned for the practical phase, ensuring that the curriculum is continually updated and meets the real needs of the company. At the coordination level, the university's academic team selects the instructors responsible for teaching the lessons, taking into account the alignment between their teaching experience and the topics defined. This composition may change between editions, due to the need to adapt to technological changes. It is up to the university to define the format of the classes, the professionals involved and the teaching approaches.

The operational level is made up of instructors with extensive academic and practical experience in the field of software development. These instructors conduct the theoretical classes in an online and synchronous format, with daily meetings in the afternoon. In addition, assistant help track the class and fostering the exchange of

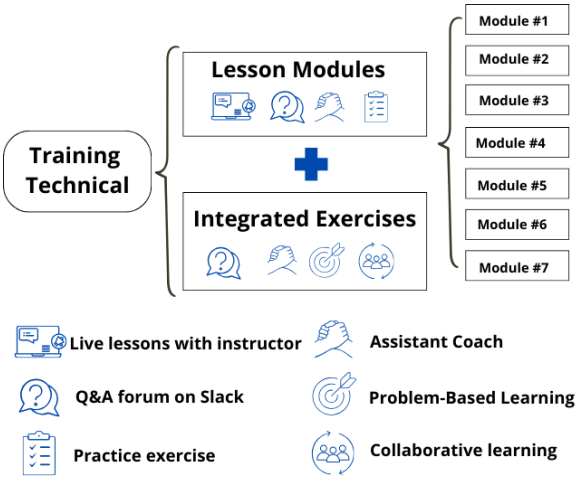

Figure 2: Technical Training Methodologies

experiences. All students maintain direct communication with the operational team through specific Slack channels, used to answer questions and share materials.

The technical training lasts approximately ten weeks, divided into seven modules. Each module covers technologies and practices which are widely used in software development at ORG. For example, the last edition contained the following modules: Programming in C#; Web development with ASP.NET MVC and Razor; Data persistence with Entity Framework; Data modeling and manipulation with SQL Server; Front-end development with HTML, CSS, JavaScript and TypeScript; Angular framework for dynamic web applications; Automated testing with Selenium.

Each module has a duration of six consecutive days. During this period, the instructors conduct three hours a day of theoretical lessons with practical contextualization, while the remaining three hours are allocated to individual exercises. These exercises encourage student autonomy, allow for the direct application of concepts and make it possible to monitor individual progress.

At the end of the seven modules, interns take part in an integrative exercise, structured to simulate a realistic software development environment. A instructor takes on the role of client and presents an informal request for a system, based on a personal need. The participants, organized into groups, have five days to plan, develop and deliver the solution, acting on their own, without prior guidance on the division of roles or methodologies to be used. This dynamic follows the principles of PBL, promoting practical research, critical thinking and team decision-making. During this activity, the mentors act only as assistants, encouraging dialog between the members of the groups. The instructors use the results of the exercise as a diagnosis of the technical and behavioral maturity of the teams, measuring skills such as self-management, collaboration and the application of acquired knowledge.

The interns come from different courses in the IT area, with varied backgrounds and experiences, and start the program with different levels of technical knowledge (different semesters of their degree course). Therefore this stage promote technical alignment

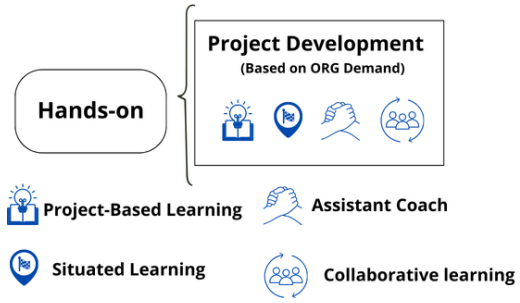

Figure 3: Hands-on Methodologies

between the participants, creating a common base of skills necessary for the start of the Hands-on stage.

5.3 Hands-on

The Hands-on is dedicated to the practical application of the knowledge acquired in the previous stages, through the development of a system based on real demands provided by the partner company, ORG. This stage adopts a Project-Based Learning (PjBL) approach, in which the project acts as the central element for mobilizing technical knowledge, planning, and decision-making (see Figure 3). The project definition including required functionalities, technologies, and infrastructure is carried out by the ORG team, creating a learning environment aligned with the principles of situated learning, where interns experience a professional development context without the risks associated with a commercial project.

In the stage the coordination level define the structure of the project and the format of the work cycles. The operational level, a instructor from the university acts as SM for the Hands-on project, guiding the teams and conducting the agile ceremonies. In addition, mentoring is incorporated through the presence of Assistant, who support the participants in the application of agile practices and the adoption of software quality principles. This mentoring structure reflects the educational role of modeling behavior and providing continuous feedback, key aspects in developing professional competencies. On the ORG operation level, the POs participate bringing demands from the company.

The development of the system acts as a guiding principle for learning, requiring students to mobilize integrated technical knowledge, planning, time management, teamwork and decision-making. The restrictions of scope, deadline and incremental delivery simulate conditions in the corporate environment. The application of Agile-Based Learning, combining Scrum and Extreme Programming (XP) practices, organizes the hands-on into four two-week cycles, called Sprints, over the course of eight weeks. At each Sprint, the participants go through a planning meeting, in which the PO presents the requirements through scenarios in the BDD format. The teams estimate effort using the Planning Poker technique and organize themselves into self-managed and self-organized groups, reinforcing the principles of collaborative learning. At each new sprint cycle, students are encouraged to reconfigure their team composition to encourage the exchange of experiences and the strengthening of collaboration. At the end of sprint, the teams hold a Sprint Demo to present the developed functionalities to the PO, followed by a retrospective to identify improvement actions for the next cycle. These reflective moments contribute to continuous learning and promote the development of autonomy and critical thinking. More details on the development framework used during the Hands-on can be found in the previous paper published [16].

Collaborative learning is further reinforced by practices such as pair programming and collective coding sessions. Throughout the process, tools such as spreadsheets, documentation, and the Burndown Chart are used to check task progress and promote team accountability. Participants experience the work environment in a way that is close to professional reality, interacting with roles, tools and dynamics similar to those found in ORG teams. Ongoing mentoring, led by SM and Assistant, focuses on developing autonomy, collaborative practice and shared responsibility among team members.

The students take responsibility for their own learning process, interacting with realistic problems, professional roles, and agile practices. By the end of the hands-on stage, participants have accumulated practical experience in software development, applying techniques and behaviors aligned with the expectations of the industry and ORG's internal standards. The educational structure of this stage serving as a pedagogical bridge between academic training and professional practice.

5.4 Internship

The final stage of the IT Program is the internship at ORG, representing the transition from the training context to the corporate environment. After completing the Hands-on stage, interns are integrated into actual development teams across different areas of the company. These teams operate with distinct technologies, scopes, and responsibilities, and the interns are assigned based on mutual alignment between team needs, technical profiles, and personal interests. This assignment process is coordinated by ORG and includes presentations from managers about available positions and areas of work.

At this stage, the interns begin to contribute to projects under the supervision of their respective teams. A key difference from previous phases is that interns are no longer working exclusively with other interns, they now collaborate with professionals of varying experience levels, including senior engineers, global colleagues, and product stakeholders. This context promotes the development of interpersonal skills, technical refinement, and adaptation to professional standards. In many cases, interns also join global teams, where interaction in English is part of daily practice, further enhancing communication skills in an international setting.

The interns' performance is continuously monitored by both the ORG and the university. Although the university assumes a more distant role during this stage, it remains available to provide support and ensure the continuity of the academic supervision until the end of the internship. Within ORG, the interns are assisted by their direct managers through regular one-on-one meetings, in which individual development goals are discussed and adjusted as needed. Each intern is also assigned a buddy, a more experienced team member who supports their onboarding, clarifies doubts, and

Table 1: Observed practices during the Selection stage of the IT Program

Lesson Learned from Selection		
Phase	**Do**	**Don't**
(i)	Apply clear eligibility rules (e.g., academic area, location, availability) to filter candidates early and improve process efficiency	Process many applications manually without automation, increasing workload in a short period
		Keep rigid selection dates regardless of applicant volume, stressing the operational team
(ii)	Use in-person tests to align with on-site work needs, reducing uncommitted candidates	Ignore the logistics of in-person tests, which require careful planning of rooms, staff, and scheduling
	Base test questions on academic and public exam sources to ensure neutrality and fairness	Disregard test anxiety, which may affect candidates even if used to written exams
	Create multiple test versions with categorized questions to ensure consistency across sessions	
	Use per-category scores instead of an overall average to better identify aligned candidates	
(iii)	Propose open problems that let candidates choose technologies and show problem-solving skills	Use only one evaluator per submission without peer checking, risking inconsistent results
	Use rubrics with clear criteria for code quality, clarity, and requirement coverage	Share vague evaluation rules, leaving candidates unsure of assessment focus
	Use tools like MOSS to ensure integrity in remote assessments	
(iv)	Include English speaking tests to complement technical evaluation and broaden profile analysis	Limit soft skills analysis to one phase and few candidates, missing strong interpersonal profiles

provides guidance during task execution. This arrangement creates a learning environment based on trust and peer mentoring, reinforcing the principles of situated and collaborative learning.

In addition to their regular team activities, interns are expected to participate in institutional initiatives that extend beyond their technical role. One of these initiatives involves joining internal committees, which are responsible for organizing lectures, workshops, volunteer actions, and social engagement campaigns within the company. Participation is encouraged both in the planning and in the execution of these events, promoting leadership, communication, and community involvement. Interns are also required to complete at least 30 hours of complementary courses of their choice throughout the internship period.

During the internship, interns can also participate in mentorship initiatives, such as the Mentor Cycle, which involves thematic discussion rounds guided by professionals from other regions and business units. These sessions offer opportunities for reflection, professional exchange, and network expansion. Furthermore, some interns take part in the rotation program, which allows them to change teams during the internship, broadening their exposure to different technologies, projects, and team cultures.

6 Lessons Learned

This section reflects on the cumulative experience gained over successive editions of the program. We presents lesson learned organized as Do's and Don't s for each stage of the program. The Do's represent aspects the authors identified as positive practices like strategies, decisions, or actions that were considered effective and potentially replicable in similar educational-industry initiatives. The Don't s, in contrast, highlight points that the authors suggest should be approached with caution, as they may be associated with limitations, inefficiencies, or challenges that could affect the program's success. This synthesis is intended to guide other initiatives in building effective and sustainable collaboration models. A key distinguishing feature of the program is the integrated development of both technical and behavioral skills, ensuring that graduates are not only proficient in technology but also prepared for the interpersonal and collaborative dynamics of industry work.

6.1 Selection Process

The observed practices during the Selection stage of the IT Program, summarized in Table 1. With an average of 800 applicants per edition for about 20 openings, the process is labor-intensive and must carefully identify candidates best aligned with the company's expectations and the internship profile.

6.1.1 Curriculum Analysis (i) Some actions were taken at the strategic level to deal with the high volume of applicants and ensure the effectiveness of the selection process. One of these initiatives was the definition of clear eligibility criteria, aligned with the requirements of the internship offered by the ORG. This initial filtering, established by the executive level, aims to prevent candidates who don't meet the prerequisites from advancing to the next stages, contributing to the efficiency of the process. Eligibility checks are a positive practice, as they allow pre-screening of candidates before the in-person test, ensuring only qualified applicants move forward. This filtering streamlines the next stages, avoids invalid

test corrections, and manages candidate expectations. However, it demands significant effort from the operational team.

6.1.2 Objective Test (ii) The decision to implement in-person stages in the selection process was aligned with the ORG internship's on-site requirement and served as a natural filter, with 20–30% of registered candidates not attending the test. Designed by instructors and assistant using public, academic sources, the exam avoids internal content to ensure neutrality and fairness. It focuses on foundational IT topics, allowing students from different academic stages to compete on equal terms. Administered in two consecutive sessions in a single afternoon, the test demands careful logistical coordination, involving multiple rooms and a sizable support team.

Good practices include categorizing questions by topic and difficulty to generate equivalent versions, reducing risks of information sharing, and using average scores per content area (rather than a single overall score) to better identify aligned candidates. However, operational challenges remain, such as managing resources and the unpredictability of attendance. While the exam may trigger anxiety due to its selective nature, the time and format mirror standard university practices, and any performance issues are more linked to emotional pressure. As part of its educational commitment, the program offers personalized feedback to all participants.

6.1.3 Technical Exercise (iii) We consider the freedom given to candidates in the technical programming exercise a valuable aspect of the selection process. The challenge presents an open-ended problem, describing only the expected functionalities of the system, without imposing specific technologies or tools. By allowing candidates to choose how to solve the problem and which technologies to use, we promote a evaluation that accommodates diverse academic and practical backgrounds. This approach encourages authentic and creative solutions and enables us to assess a broad range of skills, including front-end and back-end development, system integration, and the ability to interpret and implement requirements. Over the years, this format has proven effective in revealing candidates' problem-solving capabilities in a realistic and flexible context.

The evaluation of the technical exercise considers some aspects to promote consistency in the assessment process, a rubric was developed with predefined criteria for each of these dimensions, a practice considered positive for supporting more structured and transparent evaluations. However, the application of the rubric by a single evaluator per submission, due to time and resource constraints, can result in variations in interpretation. Additionally, the fixed schedule for test administration and correction limits the program's flexibility to adapt to fluctuations in the number of applicants. This can lead to an increased workload for evaluators and may affect the depth of the analysis. Although practices such as double-blind review could enhance the reliability of the evaluation.

As the technical programming exercise is conducted remotely, the program has limited control over task execution, which opens room for candidate collaboration or the use of external tools, such as AI. To safeguard the integrity of the process, similarity detection tools like MOSS are employed to identify potential plagiarism.

6.1.4 Interview (iv) This stage allows for the evaluation of oral communication in English, a competency valued by the ORG and not covered by the technical test, which is limited to multiple-choice questions. Assessing this skill adds depth to the process and supports a more comprehensive view of candidate profiles. However, since the soft skills assessment is concentrated in a single phase and applied to a limited number of candidates, its scope is restricted. As a result, candidates who demonstrate strong interpersonal potential but only moderate technical performance may not be identified earlier in the selection process. In general, the selection stage has proven effective in identifying candidates whose profiles align with the program's objectives and the expectations of the industry.

6.2 Technical Training

During this stage the team identified practices were identified as beneficial to the interns' development (see Table 2). One of them was the gradual progression of content, which began with introductory topics and advanced toward more complex subjects. This structure helped reduce disparities among participants from different IT programs and with varying levels of prior experience, contributing to a more balanced learning environment. Another positive aspect was the inclusion of technologies, tools, and methodologies aligned with current industry practices. Many of these topics are introduced in the later semesters of undergraduate programs or sometimes not covered at all, which made their early presentation a valuable opportunity to accelerate technical learning and better prepare interns for the practical demands of the partner ORG's projects.

The structured learning environment was strengthened by the involvement of university instructor with extensive experience in teaching computing courses. Their pedagogical background contributed to the clarity and effectiveness of the content delivery throughout the training. In addition the instructors were also accessible through an active Q&A channel on Slack, where they collaborated with assistant to respond to intern' questions in real time. This combination of experienced instruction and easily accessible support promoted ongoing communication, reinforced the instructors' role as learning facilitators, and ensured that participants received timely guidance during their learning process.

The program team observed some challenges that may indicate opportunities for improvement in future editions. The combination of a high content load within ten weeks of daily classes and the fast pace of the modules may have limited the interns' ability to fully assimilate the material. Additionally, the separation between theoretical instruction and hands-on practice may have reduced the immediate applicability of the content. These aspects appeared to have a impact on interns in the early semesters of their degree programs or with limited prior exposure to the technologies presented. The lack of formal strategies to address diverse backgrounds suggests that using teaching approaches adapted to different learning profiles could be beneficial.

Assistant suggested revising the structure of the integrative exercise. Instead of concentrating the activity at the end of all lesson completed, they proposed distributing it throughout the modules, allowing interns to develop their solutions incrementally and in alignment with the weekly content. The coordination team received the suggestion for consideration in future editions of the program.

The coordination level the team faced difficulties in composing the teaching staff for each cohort. With each new cycle, the practical projects demanded specific technologies, which required

Table 2: Observed practices during the Technical Training stage of the IT Program

Lesson Learned from Technical Training	
Do	**Don't**
Structure the training with progressive modules that balance content complexity and learning pace, supporting interns with diverse backgrounds.	Overload the training schedule without considering students' learning curves and time for reflection.
Incorporate technologies, tools, and methodologies aligned with industry practices early in the training to accelerate interns' technical development	Place the integrative exercise only at the end of the training, as this may limit opportunities to reinforce learning progressively throughout the modules.
Use instructors with prior experience teaching undergraduate students in computing courses	Ignore differences in interns' backgrounds and experience levels, as this may make it harder for some to keep up with the training.
Maintain active Q&A channels with professors and mentors to promote communication, clarify doubts, and reinforce the role of instructors as learning facilitators	Maintain a fixed teaching team without adjusting to the changing technical demands of each project cycle, as this may lead to instability and repeated reassignments.
Incorporate integrative exercises throughout the modules to promote progressive reinforcement and practical application of content.	

Table 3: Observed practices during the Hands-on stage of the IT Program

Lesson Learned from Hands-on	
Do	**Don't**
Engage interns in real company projects with internal visibility to boost motivation and professional connection	Create rigid structures or competition that block experimentation, peer learning, and socio-emotional growth
Use real demands to teach architecture, testing, version control, and collaboration through practical learning	Impose fixed workflows that hinder adaptation or innovation
Form small squads with diverse backgrounds and rotate teams each Sprint to develop communication and adaptability	Encourage competition or punish mistakes, harming a safe learning space
Foster collaboration on both front-end and back-end tasks	Exclude senior devs or tech leads, limiting feedback and deeper guidance
Guide Scrum ceremonies to reinforce purpose and structure	Assume initial training is enough; overlook continued support for advanced topics
Include a technical PO to connect interns' work with real business goals	Introduce version control tools without templates or guidance, creating barriers for newcomers
Allow autonomy in task execution with context and expectations clearly defined	Manage tasks via spreadsheets or docs, reducing traceability and complicating oversight
Give teams room to organize their workflow and propose solutions	
Use mentoring to balance autonomy with support, fostering critical thinking and ownership	
Encourage a collaborative, non-competitive culture with pair programming and shared ownership	
Treat the project as a safe space for experimentation and learning from mistakes	

the coordination team to frequently reassess the allocation of instructors. This led to instability and constant changes in both the teaching staff and the content offered. This scenario highlighted the importance of maintaining a network of instructors with diverse technical profiles and openness to adapt to ongoing changes.

6.3 Hands-on

The hands-on stage provided some observations about practices that appeared to promote positive outcomes, as well as aspects with potential for improvement (see Table 3). Structuring the Hands-on project around a real demand from ORG seemed to support intern engagement throughout the practical stage. Direct monitoring by company professionals and the presentation of deliverables to managers at the end of each Sprint appeared to encourage greater commitment to the quality of the developed solutions. Based on the experience, projects with a concrete purpose and internal visibility within the company were perceived as factors that may contribute

Table 4: Observed practices during the Internship stage of the IT Program

Lesson Learned from Internship	
Do	**Don't**
Integrate interns into real development teams, with active participation in ongoing projects and exposure to production contexts	Assign interns to teams without ensuring alignment between their profiles and the team's current scope or dynamics
Maintain structured support through buddy systems, mentoring cycles, and one-on-one meetings with managers to foster development	Rely exclusively on informal feedback; the absence of structured follow-up may limit learning and engagement
Include interns in global teams to promote communication in English and intercultural interaction	Overlook language challenges or cultural barriers that may hinder full participation in international teams
Encourage participation in rotation programs to allow exploration of different teams, technologies, and work cultures	Neglect opportunities for integration into the company's broader culture and non-technical development spaces
Foster engagement beyond technical tasks through participation in internal committees and corporate initiatives	Delegate course planning entirely to interns without guidance, which may result in poor or misaligned training choices
Require a minimum workload of complementary courses while supporting interns in choosing relevant paths for their career goals	

to increased student motivation and a stronger connection to the professional environment.

The formation of small squads, composed of approximately five interns with mixed technical backgrounds, was perceived as a strategy that contributed positively to the team dynamics. The distribution of participants with varying levels of knowledge appeared to support peer learning and collaborative development of functionalities, without strict divisions between front-end and back-end roles. Additionally, the planned reorganization of teams at each Sprint seemed to encourage adaptability to different work styles, while promoting communication, flexibility, and collaboration across diverse group settings. Another aspect perceived as positive was the structured and guided use of Scrum ceremonies. Assistant Coaches, SM, and POs provided support throughout these activities, combining practical execution with moments of methodological explanation. The ceremonies were conducted with a focus on helping interns understand the purpose, structure, and value of each step, adapting the guidance according to the group's needs. The PO's active involvement, supported by technical expertise and availability for frequent interaction, helped clarify the demands and actively encouraged interns to participate in development-related decisions.

The mentoring model adopted allowed interns to act with autonomy while receiving contextualized support when needed. This approach encouraged them to take responsibility for how tasks were executed and to organize their own work methods. Interns were also encouraged to propose alternative solutions based on their understanding of the context, which contributed to the development of critical thinking and decision-making skills. Practices such as pair programming and collective development sessions promoted knowledge exchange among peers and supported the development of both technical and interpersonal competencies.

The collaborative learning environment, free from competition and focused on shared deliverables, was seen as a factor that encouraged experimentation and personal growth. Structuring the project in multiple sprints with incremental deliveries and continuous guidance enabled interns to progressively refine their practices and behaviors. The team also identified the value of maintaining flexibility during the process, adapting explanations, examples, and strategies according to the group's evolving needs. This responsiveness reinforced the active and student-centered nature of the training. The diversity of intern profiles contributed to the development of socio-emotional skills relevant to collaborative work.

The management team also observed technical and organizational aspects that have required ongoing reflection and adjustment. One point under consideration has been the lack of more experienced professionals, such as senior developers or tech leads with direct access to the repository. This has raised discussions about the importance of expanding formal code review mechanisms. In response, the team has been encouraging peer review practices and reinforcing the technical support offered by assistant. Although the technologies used in the project are addressed during the training phase, it was noted that some interns begin the practical stage with limited knowledge of certain topics. In this context, the need for continuous technical follow-up during development is being considered, especially to support the deepening of more advanced topics related to code quality and good development practices.

Another aspect identified was the difficulty reported by some interns in using version control tools, particularly regarding branch creation, segmented commits, and writing merge request descriptions. Based on these observations, the operation team started developing templates and guidance materials to support these activities. Additionally, the possibility of integrating a task management tool, such as Jira, is being explored to improve tracking of activities and deliverables, overcoming limitations encountered when using spreadsheets and shared documents.

The lack of structured automated testing practices and continuous integration and delivery (CI/CD) pipelines has also been perceived as a factor that reduces visibility into application behavior and limits early error detection. As a response, squads have been encouraged to gradually implement simple automated tests and pipelines as part of their deliveries, even at an introductory level. From a behavioral standpoint, occasional difficulties in team interaction were observed. The formation of affinity subgroups or

the rigid division of tasks based on technical preferences has highlighted the need for clearer strategies for conflict mediation and the promotion of multidisciplinary collaboration.

6.4 Internship

The internship stage marks the transition from a structured training environment to real-world team integration, where interns face actual demands and responsibilities. One of the key strengths is the opportunity to work directly within company teams, contributing to ongoing projects while being guided by experienced professionals. Structured support mechanisms such as regular one-on-one meetings with managers, buddy assignment, and participation in global mentoring initiatives help sustain the interns' development and adaptation during this transition (see Table 4).

The inclusion of interns in global teams fosters language skills and intercultural competence, while the rotation program offers exposure to diverse technological domains, helping interns identify areas of greater alignment with their professional interests. These strategies contribute to professional growth, autonomy, and adaptability. However, certain aspects of this phase may require refinement. The autonomy granted to interns in managing their complementary learning, such as the selection of optional courses, can sometimes result misalignment with their actual development needs. Additionally, the strong social bonds formed during earlier stages can create resistance to broader integration into company teams. Finally, while the interns are expected to deliver real outcomes, it is important to ensure that learning remains central, with guidance and support balanced against performance expectations.

7 Conclusion

This experience report presents a software engineering training program for undergraduate students in information technology, conducted through a partnership between the University and a multinational IT company. Created to meet the demand for professionals ready to work on real industry projects, the program has been running for over 20 years, with 21 cohorts completed. Since its current structure in 2016, it has been held biannually, with around 20 participants per cohort. The program is annually reviewed to adapt to the partner company's needs, with a multidisciplinary team implementing necessary changes. Performance monitoring continues after completion, with a 96% success rate and about 15% of former interns advancing two to three levels in under a year.

The program is structured in four phases: selection, technical training, hands-on practice and an internship within the company. Selection, with multiple elimination stages, helps to identify students who match the technical and behavioral profile expected by the company. The technical training, conducted by university professors, introduces tools and technologies used in industry. In the hands-on, the students work on real projects, organized into self-managed teams, with continuous monitoring by mentors and the application of agile practices, encouraging the development of technical, collaborative and decision-making skills.

The results show that the program plays an important role in training software engineering talent by offering a training path, from structured selection to full integration into professional teams. For the university, the program represents a concrete opportunity to innovate in teaching and provide practical experiences that complement traditional training. For the company, it is an effective strategy for developing talent in line with its technical, cultural and organizational needs.

The experience accumulated over the years allows the model to be continuously improved, and also provides valuable input for institutions and companies interested in structuring similar partnerships. The systematization of the lessons learned in each phase, through the identification of successful practices and points of attention, contributes to the construction of more effective, sustainable programs aligned with the challenges of software engineering.

ARTIFACT AVAILABILITY

Due to NDA constraints, it is not possible to provide a replication package containing the materials created and used during the training program. The authors are available for contact regarding any clarifications.

ACKNOWLEDGMENTS

Natalya Goelzer and Pedro Portella thank Dell Brazil for the financial grants sponsored using incentives from the Brazilian Informatics Law (Law no. 8.248/1991). Sabrina Marczak thanks CNPq for the financial support through a Productivity Scholarship (process no. 313181/2021-7).

REFERENCES

[1] Aline Andrade, Regina Albuquerque, Tania Dors, Fabio Binder, Andreia Malucelli, and Sheila Reinehr. 2021. Teaching, Innovation, and Software Development: The Use of Reflective Practice. In *Proceedings of the XII Brazilian Congress on Software: Theory and Practice (CTIC-ES)* (Joinville, Brazil). Brazilian Computer Society (SBC), Porto Alegre, RS, Brazil, 137–138. https://doi.org/10.5753/cbsoft_estendido.2021.17299

[2] Lilian Bacich and José Manuel Moran. 2018. *Active Methodologies for Innovative Education: A Theoretical and Practical Approach.* Penso, Porto Alegre, Brazil.

[3] Maria Ivanilse Calderon Ribeiro and Odette Mestrinho Passos. 2020. A Study on the Active Methodologies Applied to Teaching and Learning Process in the Computing Area. *IEEE Access* 8 (2020), 219083–219097. https://doi.org/10.1109/ACCESS.2020.3036976

[4] Ana Clementino, Erick Lima, Luann Lima, André Guedes, Dorgival Netto, and Jarbele Coutinho. 2024. Teaching Software Engineering: An Overview of Current Approaches and Practices in the Last Decade of SBES. In *Anais do XXXVIII Simpósio Brasileiro de Engenharia de Software* (Curitiba/PR). SBC, Porto Alegre, RS, Brasil, 422–432. https://doi.org/10.5753/sbes.2024.3517

[5] Thelma Colanzi, Leandro Silva, Andressa Medeiros, Paulo Gonçalves, Eniuce Souza, Douglas Farias, and Greicy Amaral. 2023. Practicing the Extension in Software Engineering Education: an Experience Report. In *Proceedings of the XXXVII Brazilian Symposium on Software Engineering* (Campo Grande, Brazil) *(SBES '23)*. Association for Computing Machinery, New York, NY, USA, 514–523. https://doi.org/10.1145/3613372.3614196

[6] Thais E. Colanzi et al. 2023. Project-Based Learning in Graduate Software Engineering Courses. In *SBES Education Track*.

[7] Allan Collins, John Seely Brown, and Susan Newman. 1989. Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In *Knowing, learning, and instruction: Essays in honor of Robert Glaser*, Lauren B. Resnick (Ed.). Lawrence Erlbaum Associates, 453–494.

[8] Aldo Dagnino. 2014. Increasing the effectiveness of teaching software engineering: A University and industry partnership. *IEEE 27th Conference on Software Engineering Education and Training (CSEET), Klagenfurt, Austria* (2014), 49–54. https://doi.org/10.1109/CSEET.2014.6816781

[9] Marian Daun, Jennifer Brings, Marcel Goger, Walter Koch, and Thorsten Weyer. 2021. Teaching Model-Based Requirements Engineering to Industry Professionals: An Experience Report. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. 40–49. https://doi.org/10.1109/ICSE-SEET52601.2021.00013

[10] Pierre Dillenbourg, Michael J. Baker, Agnès Blaye, and Claire O'Malley. 1995. The evolution of research on collaborative learning. In *Learning in Humans and*

Machine: Towards an Interdisciplinary Learning Science, Elisabeth Spada and Paul Reiman (Eds.). Elsevier, Oxford, 189–211.

[11] Luciana Mara Diniz, Fischer Ferreira, and João Paulo Diniz. 2021. Interdisciplinarity in Teaching Software Engineering and Human-Computer Interaction Using Digital Technologies: An Experience Report. In *Proceedings of the XXVII Workshop on Informatics in Education (WIE)* (Online). Brazilian Computer Society (SBC), Porto Alegre, RS, Brazil, 116–127. https://doi.org/10.5753/wie.2021.218683

[12] Peggy A. Ertmer and Timothy J. Newby. 2013. Behaviorism, Cognitivism, Constructivism: Comparing Critical Features From an Instructional Design Perspective. *Performance Improvement Quarterly* 26, 2 (2013), 43–71. https://doi.org/10.1002/piq.21143

[13] José Figuerêdo, Jussara Machado, Samuel Lima, Cláudio Cerqueira, and Claudia Pereira. 2021. The Experience of Tutoring in Algorithms and Programming Courses in Engineering from the Perspective of the Tutors. In *Proceedings of the Brazilian Symposium on Computing Education (EduComp)* (Online). Brazilian Computer Society (SBC), Porto Alegre, RS, Brazil, 183–192. https://doi.org/10.5753/educomp.2021.14484

[14] Paul Gestwicki and Khuloud Ahmad. 2011. App Inventor for Android with Studio-Based Learning. *Journal of Computing Sciences in Colleges, Evansville, USA* 27, 1 (oct 2011), 55–63.

[15] Robert Glaser. 1991. The maturing of the relationship between the science of learning and cognition and educational practice. *Learning and Instruction* 1, 2 (1991), 129–144. https://doi.org/10.1016/0959-4752(91)90023-2

[16] Natalya Marjana Goelzer, Pedro Portella Possamai, Sabrina Marczak, Michael da Costa Móra, and Daniel Antonio Callegari. 2024. Nurturing Talent: The IT Academy Journey into Quality Development. In *XXIII Brazilian Symposium on Software Quality (SBQS '24)*. Association for Computing Machinery, New York, NY, USA, 508–518. https://doi.org/10.1145/3701625.3701636

[17] Alexandre Grotta and Edmir P. V. Prado. 2018. An Essay on the Educational Experience in Computer Programming: The Traditional Approach versus Project-Based Learning. In *Proceedings of the XXVI Workshop on Informatics Education (WEI)* (Natal, Brazil). Brazilian Computer Society (SBC), Porto Alegre, RS, Brazil. https://doi.org/10.5753/wei.2018.3496

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

[19] Maurício Kalinowski et al. 2023. Does Higher Education Meet Industry Demands?. In *Proceedings of the Brazilian Symposium on Software Engineering (SBES)*.

[20] Maurício Kalinowski et al. 2023. An Industry–University Distance Education Program for Software Engineering Specialization. In *Proceedings of the Brazilian Symposium on Software Engineering (SBES)*.

[21] Marcos Kalinowski, Tatiana Escovedo, Fernanda Pina, Adriana Vidal, Ariane Pereira Da Silva, Ricardo Ponsirenas, and Daiana Garibaldi Da Rocha. 2023. Training the Professionals that Industry Needs: The Digital Software Engineering Education Program at PUC-Rio. In *Proceedings of the XXXVII Brazilian Symposium on Software Engineering* (Campo Grande, Brazil) *(SBES '23)*. Association for Computing Machinery, New York, NY, USA, 37–46. https://doi.org/10.1145/3613372.3614200

[22] Marco Kuhrmann, Joyce Nakatumba-Nabende, Rolf-Helge Pfeiffer, Paolo Tell, Jil Klünder, Tayana Conte, Stephen G. MacDonell, and Regina Hebig. 2019. Walking Through the Method Zoo: Does Higher Education Really Meet Software Industry Demands?. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. 1–11. https://doi.org/10.1109/ICSE-SEET.2019.00009

[23] Jean Lave and Etienne Wenger. 1991. *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press, Cambridge.

[24] Christina P. Lee et al. 2021. Coaching Professional Software Developers. In *Proceedings of the ICSE Software Engineering in Practice (SEIP)*.

[25] André M. Lima and Liane T. Silva. 2020. Aprendizagem situada no ensino de engenharia de software: uma experiência com projetos reais. In *Proceedings of the Brazilian Congress on Software Engineering Education (CBEES)*. 112–123.

[26] Lyziane Nogueira, Zezineto Segundo, Sebastião Alves Filho, Jéssica Araújo, Rommel Lima, and Ceres Morais. 2024. The Role of Academic Tutoring in the Teaching and Learning Process of Programming: An Experience Report. In *Proceedings of the XXXII Workshop on Informatics Education (WEI)* (Brasília, DF, Brazil). Brazilian Computer Society (SBC), Porto Alegre, RS, Brazil, 285–296. https://doi.org/10.5753/wei.2024.2961

[27] Frederico Sauer Guimarães Oliveira, Yuri de Abreu de Melo, and Martius Vicente Rodriguez Y Rodriguez. 2023. Motivation: A Challenge in the Application of Active Methodologies in Higher Education. *Avaliação: Journal of Higher Education Assessment* 28 (2023), e023004. https://doi.org/10.1590/S1414-40772023000100004

[28] Rafael Oliveira and S. Marczak. 2023. Experience Report on Software Engineering in the Computer Science Curriculum. In *SBES Education Track*.

[29] Rafael Prikladnicki, Adriano Bessa Albuquerque, Christiane Gresse von Wangenheim, and Reinaldo Cabral. 2009. Software Engineering Education: Challenges, Teaching Strategies, and Lessons Learned. In *Proceedings of the Brazilian Forum on Software Engineering Education (FEES)*. Brazil, 1–12.

[30] Mário Rebouças et al. 2022. Content and Quality of Sprint Retrospectives in Software Engineering Education. In *Proceedings of the ICSE Software Engineering Education and Training (SEET)*.

[31] Cynthia Pinheiro Santiago, José Wally Mendonça Menezes, and Francisco José Alves de Aquino. 2023. Proposal and Evaluation of a Project-Based Learning Methodology in Software Engineering Courses through a Didactic Sequence. *Brazilian Journal of Computers in Education* 31, 1 (Feb. 2023), 31–59. https://doi.org/10.5753/rbie.2023.2817

[32] Cynthia Pinheiro Santiago, José Wally Mendonça Menezes, and Francisco José Alves de Aquino. 2023. Proposal and Evaluation of a Project-Based Learning Methodology in Software Engineering Courses Through an Instructional Sequence. *Brazilian Journal of Computers in Education* 31, 1 (feb 2023), 31–59. https://doi.org/10.5753/rbie.2023.2817

[33] Thais Silva, Gláucia Braga e Silva, and Maria Theresa Henriques. 2021. Mentorship in Programming: Learning by Teaching and Teaching by Learning. In *Proceedings of the XV Women in Information Technology (WIT 2021)* (Online Event). Brazilian Computer Society (SBC), Porto Alegre, RS, Brazil, 310–314. https://doi.org/10.5753/wit.2021.15872

[34] George Skevoulis. 2021. Engineering a Successful Partnership Between Academia and the Financial Industry. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE)*.

[35] Milene Souza et al. 2024. IT Talent Shortage: Strategies to Mitigate a Blackout. In *Proceedings of the Brazilian Symposium on Software Engineering (SBES)*.

[36] Maurício Souza, Renata Moreira, and Eduardo Figueiredo. 2019. Students Perception on the use of Project-Based Learning in Software Engineering Education. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering* (Salvador, Brazil) *(SBES '19)*. Association for Computing Machinery, New York, NY, USA, 537–546. https://doi.org/10.1145/3350768.3352457

[37] Tiago Souza et al. 2023. Impact of Industry Partnerships on PBL Initiatives. In *Proceedings of the Brazilian Symposium on Software Engineering (SBES)*.

[38] Caio Steglich, Anielle Lisboa, Rafael Prikladnicki, Sabrina Marczak, Michael da Costa Móra, Alejandro Olchik, Nelice Heck, Yasser Rachid, and Guilherme Ghidorsi. 2020. Agile Accelerator Program: From Industry-Academia Collaboration to Effective Agile Training. In *Anais do XXXIV Simpósio Brasileiro de Engenharia de Software, Natal, Brasil* (Natal, Brazil) *(SBES '20)*. Association for Computing Machinery, New York, NY, USA, 21–30. https://doi.org/10.1145/3422392.3422485

[39] Simone Tonhão, Andressa Medeiros, and Jorge Prates. 2021. Uma abordagem prática apoiada pela aprendizagem baseada em projetos e gamificação para o ensino de Engenharia de Software. In *Anais do Simpósio Brasileiro de Educação em Computação* (On-line). SBC, Porto Alegre, RS, Brasil, 143–151. https://doi.org/10.5753/educomp.2021.14480

[40] Jos van der Linden, Gijsbert Erkens, Henk Schmidt, and Peter Renshaw. 2000. *Collaborative Learning*. Springer Netherlands, Dordrecht, 37–54. https://doi.org/10.1007/0-306-47614-2_3

[41] J. Vanhanen and H. Korpi. 2007. Experiences of Using Pair Programming in an Agile Project. In *Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS)*. IEEE, Hawaii, USA, 274b–274b. https://doi.org/10.1109/HICSS.2007.448

[42] Laurie Williams, Robert R Kessler, Ward Cunningham, and Ron Jeffries. 2000. Strengthening the case for pair programming. *IEEE software* 17, 4 (2000), 19–25.

[43] Diego Teixeira Witt and Avanilde Kemczinski. 2020. Active Learning Methodologies Applied to Computing: A Literature Review. *Informatics in Education: Theory and Practice* 23, 1 Jan/Apr (May 2020). https://doi.org/10.22456/1982-1654.90319