An undergraduate Software Engineering practice course: bridging the academia-industry gap

Sofia Larissa da Costa Paiva
Instituto de Informática, Universidade Federal de Goiás – UFG
Goiânia, Brazil
sofialarissa@ufg.br

Adriana Silveira de Souza
Juliano Lopes de Oliveira
Instituto de Informática, UFG
Goiânia, Brazil
adriana.silveira@ufg.br
julianolopes@ufg.br

Mariana Soller Ramada
Murilo Lopes da Luz
Instituto de Informática, UFG
Goiânia, Brazil
marianaramada@ufg.br
muriloluz@ufg.br

ABSTRACT

Teaching problem: Software Engineering (SE) practice comprises a set of activities that Software Engineers perform in their typical professional work to produce solutions for needs arising from stakeholders. To learn these activities, the undergraduate student must leave the controlled academic environments that deal with isolated and simplified problems using a single course point of view to reach concrete world working environments, where complex issues must be solved to address the conflicting interests of several stakeholders. **Research question**: What approaches to undergraduate courses would provide adequate conditions for students to learn SE practice? **Situating the case**: This paper reports experiences from applying an approach to teaching SE practice in an undergraduate course using a typical industry scenario, where complex SE projects are executed to meet the needs of actual stakeholders. **How this case was studied**: The reported experience shows the evolution of the proposed course approach after its application on three academic semesters in an undergraduate SE Bachelor's Degree program in a public university in Brazil. The research material for this report was informally collected over three years of teaching the course, from 2021 to 2023, and indicates that the course approach promotes students' professional attitudes and technical competencies.

CCS CONCEPTS

• **Social and professional topics** → **Employment issues**; **Software engineering education**.

KEYWORDS

Software Engineering Education and Training, Teaching Software Engineering Practice, Undergraduate Software Engineering Course Practice

1 INTRODUCTION

Software Engineering (SE) practice involves performing typical software development and sustainment activities to provide products and services that address stakeholder's requirements and expectations [25]. In the context of a SE higher education program, there are two primary environments where these practical activities can be carried out: academic laboratories and software production units.

Academic laboratories are controlled environments where practical SE learning activities are carried out based on treating simplified problems and, generally, from the perspective of a single course. In this context, practical activities focus on teaching/learning specific course topics without generating products or services that could be useful to any interested party. On the other hand, software production units are industrial environments where SE projects are carried out professionally to develop or sustain actual software products or services that meet needs and expectations of project's stakeholders. Although academic laboratories attempt to prepare professionals for software production units, there is a gap between industrial needs and software practitioners' education in terms of specialized software tools, testing, security, and soft skills [3, 31, 32].

The SE undergraduate program that provides the context for the experiences described in this paper includes students' activities within both environments. Practical SE activities are continuously carried out from the beginning of the curriculum in the context of traditional courses that compose the SE Bachelor's Degree program, such as software design and construction courses. These activities are conducted in controlled academic laboratories and deal with simplified problems, focusing on the specific course point of view. For instance, laboratory activities for software construction course assume that requirements and design issues are already resolved, so the practice deals exclusively with code implementation and integration. To improve their professional maturity, the students complete their practical training with a one academic semester immersion in a specific software-producing unit in the context of a particular course called ***Software Engineering Practice*** (SEP), which is taught in the last semester of the SE degree program. In this course, students learn SE practice and professional attitudes by acting in project roles in a genuine software-producing facility that deals with actual SE problems to develop, deliver, and maintain software that addresses external stakeholders' requirements.

This article reports experiences on conducting this SEP course during three academic semesters in a Brazilian public university. We have applied the Action Research Methodology [45], based on a dialogue space where the involved actors collaborate in solving problems, proposing solutions, and learning in action. The principal research question was: *What approaches to undergraduate courses would provide adequate conditions for students to learn SE practice?* The SEP course, described in the remainder of this paper, is a possible answer to this question. Section 2 discusses related work and key concepts for the SEP course. Section 3 presents an overview of the SEP course approach. Section 4 describes the evolution of the course methodology. Section 5 analyses the SEP course from a SE competencies perspective. Section 6 summarizes lessons learned from the academic experiences of teaching SEP. Section 7 concludes with a synthesis of SEP course contributions to improvement of students' professional maturity.

2 BACKGROUND AND RELATED WORK

Preparing individuals to be SE professionals in real-world conditions is a well-known issue, and it still challenges the SE community [22, 28, 32, 36]. An effective approach to teaching SE practice must prepare individuals to be productive for companies as soon as possible and deliver graduates aligned with the industry's expectations. This is a great challenge involving the misalignment problem of academia and industry SE competencies points of view and expectations [8, 10, 11, 18, 35, 41]. One key question is whether universities have the resources to simulate a realistic SEP environment or whether software companies should provide such conditions themselves. The ideal situation combines university education and company training to develop SE competencies [13] and is the approach followed in the present work.

2.1 SE curriculum and competency

Traditional education involves expository classes to transfer knowledge and a few courses to learn how to apply the knowledge [14]. However, modern curriculum guidelines for SE undergraduate courses emphasize the need to move beyond the theoretical class towards a competency-based education [2, 4, 23, 40, 42]. The ACM and IEEE Computing Curricula 2020 (CC2020) [2] sets out guidelines for the global definition of computing teaching in a wide range of disciplines, such as computer science, information systems, and software engineering, providing requirements for structuring the curricula of these disciplines. CC2020 has shifted the vision from learning knowledge to developing competencies. In this sense, a better understanding of the SE competency construct enables the advancement of SE education, speeding up the training of students and professionals to meet software industry needs [6].

Among the elements contributing to competency, knowledge, skill, and attitude are primarily consensual. Knowledge comprises information, concepts, ideas, methods, and procedures that a person has acquired through their education, training (learning), and experience. Skill deals with how knowledge can be applied to a given context. Attitude is related to action, manifested by the desire to do, and establishes an individual's relationship with a concept or object, which can be positive or negative.

However, SE competency is a multidimensional construct with many other elements and interpretations according to different theoretical schools of thought on the concept of competency [17]. For instance, American schools associate competency with knowledge and cognitive ability. On the other hand, for French schools, competency must be demonstrated in the form of a product or service that meets the needs and requirements of actual stakeholders. The approach described in this paper complies with both schools. Technical knowledge is essential for SE competency and is based on SWEBOK knowledge areas [25]. At the same time, the cognitive and emotional skills were taken from the mental abilities and behavioral skills and attributes cited in SWECOM [26], from the soft skills identified by Caieiro-Rodrígues et al. [9] and from self and social competencies defined by Thurner and Böttcher [46]. The assessment of competencies is based on observable results of the student's performance on SE projects.

2.2 Teaching SE practice

Active methodologies, such as problem-based learning (PBL) and learning by doing (LBD), are the most used by instructors in SE undergraduate courses to bring practical experience for students [34]. These methodologies are simple to apply and do not require advanced instructors' capabilities. However, approaches involving real projects and clients are challenging since they are more complex and demand significant effort from instructors, teaching assistants, and sometimes clients and users [34]. Artificial problems are usually adopted, which hinders the primary goal of applying students' competencies to solve genuine software industry problems.

Team *capstone projects* are another popular way to provide students with experience beyond theoretical classes [7]. Students can be organized into teams to plan and develop a software product from start to finish during an academic year, integrating concepts learned in several different subjects [2]. Applying the development of capstone projects as an educational tool is helpful because it enables the knowledge and skills internalization in practice [43]. Since such a project is usually right before students enter the industry, some universities heavily emphasize demands from the industry as capstone projects. Some demands typically not practiced in theoretical courses include [30]: focus on integrating technical skills learned in isolation and gaining experience in soft skills such as leadership. One method universities adopt is partnering with industry to serve as clients for capstone projects [7, 24, 38, 39]. However, integrating these experiences into the courses is a complex challenge for universities [34]. Significant effort is necessary regarding time and human resources to support the instructional process, which is challenging for most universities. Instructors with software development expertise, who frequently require real clients who act as counterparts, are necessary for these projects [47].

Serious games and simulation are proposed alternatives to reduce the cost of teaching SE practice [20, 21]. However, such approaches also face significant challenges, mainly from the educator's point of view [15]. For instance, games may enhance learning but also may distract students from the actual learning goals. Moreover, designing games is expensive and complex work, and poorly designed games can hinder learning instead of improving it.

Maguire et al. [33] proposes a cooperative approach to software engineering education for academia and industry to partner to deliver graduate professionals. This approach attempts to address the isolation of theory and practice by integrating them in a single program where students are exposed to theory at university and practice it in the workplace. In particular, the academic and industrial partners work together to form a program that balances theory and practice. Sabariah et al. [39] also proposes an industry-academia collaboration program as an opportunity for undergraduate students to learn technologies, do an internship in an actual project with real problems, and, in the end, present their work as a capstone project. Ohlsson and Johansson [36] describes a two-year program that complements four-and-a-half-year degree programs at traditional engineering schools. Since the target students have yet to gain a background in SE, the program practice aims to learn the foundations of this discipline to complement the student's formation. Our proposal takes only one academic semester and focuses

on developing competencies previously learned in a complete SE undergraduate program.

Souza et al. [12] presents a systematic literature mapping that identifies only nine primary studies on teaching approaches applied to software factories in information technology higher education programs. The primary teaching approach in these studies is PBL, and the systematic review conclusions point out the need for processes, techniques, and patterns to guide the adoption of a software factory in SE educational programs. This paper reports experiences that contribute to satisfying these needs.

Our approach differs from previous work mainly because in the SEP course students work in an industrial context with external clients' demands. PBL and capstone projects are also student-centered methodologies in which students seek a solution to a certain problem, however, they are applied in the context of a specific course (for example, Software Design), focusing on a particular aspect of software engineering, and having no interaction with external clients. Moreover, there is no industrial environment in which students need to demonstrate technical knowledge and in addition follow processes and rules, work in a team, respect leadership, and deal with real world constraints. Students on internship can work in an industry environment, but the assigned tasks are too simple, there is little interaction with the experienced staff, and there is no educational feedback to guide on competencies evolution. Our approach aims at authentic learning, as defined in [37], situating students in a learning context with real world problems, which students are likely to face in their real professional career. Students work closely with experienced technical leaders, and exercise soft skills that would not be relevant in an academic environment or in individual work.

3 SEP COURSE THEORETICAL FRAMEWORK

The SEP course deals with practical activities carried out by SE students in the context of a software-producing organization, according to the following syllabus:

> *Application of the SE body of knowledge in the context of projects in a software-producing organization. Use of SE processes in scope and depth. Selection and use of SE standards, methods, techniques, and tools to achieve objectives established in projects. Integration and consolidation of knowledge and skills expected from SE professionals. Exercise of professional practices and attitudes based on the SE code of ethics and professional posture. Practice in SE technical processes. Practice in SE management processes. Practice in SE technologies.*

To enroll in a SEP course, the student must have succeeded in all other courses in the SE educational curriculum. This last stage of the degree program is characterized by the concrete opportunity for integrating theory and practice, involving problem situations generated by field experience that lead to periodic research activities, consultancies, debates, and students' adoption of new behaviors.

3.1 SEP Objectives

The SEP course aims to improve (a) students' maturity in SE practical activities and (b) student's professional practice and ethical behavior. Three main categories of SE processes are addressed in a SEP course: technical, managerial, and technological. All activities must be fully committed to the SE code of ethics [1].

Practice within technical processes exercise and expand skills in the technical SE processes group [27], in an environment as close to the typical software workplace as possible, to provide conditions for students to:

- Acquire a broad and integrated view of SE technical processes, developing a solid perception of how to use technical skills ethically and effectively.
- Review and use technical SE processes, integrating technical knowledge acquired in the course and understanding the relationship between them and their importance for building and sustaining high-quality software.
- Carry out technical activities related to: systems and software requirements engineering, configuration management, architecture, modeling, design, verification and validation, construction, integration, and maintenance; software reading, understanding, and testing; software and information security, usability, compatibility, performance efficiency, reliability, and protection; and data modeling and persistence.

Practice within management processes exercise and expand skills in managerial and organizational SE activities to:

- Acquire a broad and integrated view of SE managerial and organizational processes, developing a solid perception of how to use management skills ethically and effectively.
- Review and use SE management processes, integrating managerial knowledge acquired in the course and understanding the relationship between them and their importance for building and sustaining high-quality software.
- Carry out activities related to ethical and professional management principles of SE, such as: process and software product quality; project management and measurement; software contracting and economics; management of people and software teams; governance and management of software services; software and information security; software research and experimentation methodologies.

Practice within SE Technologies exercise and expand skills in applying SE tools and frameworks to:

- Acquire a broad and integrated view of SE tools' technologies, developing a solid perception of how to use these technologies ethically and effectively.
- Review and use SE tools, integrating technological knowledge acquired in the course and understanding the relationship between them and their importance for building and sustaining high-quality software.
- Carry out activities related to CASE and IDE tools, such as: database management systems; web application servers; programming and scripting languages; tools for modeling and generating user interfaces; code and document versioning; environment and project repositories; scheduling and task management; software debugging and testing; code building and integration; code quality analysis.

After a successful accomplishment of SEP course, students are expected to develop and improve the following broad SE competencies while performing technical and managerial roles in projects:

(1) Carrying out SE tasks according to project objectives, applying the SE body of knowledge to its fullest extent to generate high-quality software products or services.
(2) Applying SE competencies, with scope and depth, appropriately to various situations in software projects.
(3) Planning and executing software projects, reconciling stakeholder needs, project objectives, and real-world constraints in which the software operates.
(4) Planning, carrying out, and appropriately modifying products and services relevant to organizational SE processes.
(5) Acting, individually and in teams, according to the SE code of ethics, exercising appropriate SE professional attitudes, inside and outside the project environment.
(6) Modifying, evolving, verifying, and validating artifacts resulting from SE processes created by third parties.

3.2 SEP environment: the Software Factory (SF)

The Software Factory (SF) is a software production unit within the university. It provides and maintains, within an academic environment, a professional infrastructure for producing and sustaining software, combining software industry's best practices with theories and innovations generated by the university. The SF permanent team comprises one director and five full-time SE professionals.

The main objective of SF is to provide efficient, innovative, and high-quality solutions for needs involving research, development, evaluation, or application of software processes and technologies, promoting integration between teaching, research, and outreach activities at the university. [1]. SF plans, manages, operates, maintains, and evolves a suitable technological environment for producing and sustaining high-quality software to fulfill this objective. This environment promotes the qualification of project participants, which is also an objective of the SF: to contribute to the maturity and improvement of software professionals.

SF works with multidisciplinary technical teams guided by institutionalized software processes and policies. Despite being located in an academic environment and having students in the composition of some teams, the work at the SF is guided by policies and processes that govern how everyone involved works. One of SF's management processes involves monitoring project members to ensure everyone, including students, follows the same processes. The SF environment is a suitable setting for practical activities in the professional training of Software Engineers because it is a representative instance of the regional software industry. However, it is located in an academic organization. SF receives demands for software products and services from the academic community and society. These demands involve different types of software and require different SE competencies, such as software requirements, architecture, design, construction, integration, testing, deployment, migration, and maintenance.

The design and creation of SF were performed in conjunction with the undergraduate course in SE since the SF was conceived as one of the pillars for the SE curriculum pedagogical project, which specifies that *"the SE Practice course is completed with 320 hours of activities related to SF projects"*. The commitment is corroborated by the internal regulations of SF, which establish the integration of practical activities defined in the course as a responsibility of the SF director. This commitment between the course and SF ensures that the SF's Portfolio Management process prioritizes the execution of projects that present adequate conditions for the exercise of practical SE activities by undergraduate students.

SEP professors and students are designated to work on SF's projects each academic semester. All SEP activities are carried out in the context of SF's projects under the direct supervision of the professors responsible for the course. Thus, a SEP course requires prior planning of projects at SF, taking into account the pedagogical aspects inherent to the participation of students in these projects. In this way, the planning of each SEP course section must be aligned with the planning of SF's project portfolio. The SF director, executing the Project Portfolio Management Process, decides, considering the available resources, the most appropriate time to start each project. Therefore, there may be projects being carried out at SF without the participation of students or professors, as SF has its collaborator staff. SEP students are extra human resources available to SF during an academic semester. SF projects are not restricted to the academic term; students only participate in projects during this term, following the academic calendar. Thus, SF projects follow their schedules and have no restrictions related to academic calendar; however, there are human resources (SEP students and professors) with this restriction available for some SF projects.

The project manager must be an SF collaborator to ensure that every project follows SF's policies and processes and has a full-time manager, as recommended by best practices in project management. When a project involves an SEP course instance, the project management team comprises professors and the project manager. This team must support the project manager's decisions in planning and monitoring the project's actions. In particular, the allocation of students to project activities is decided by the whole management team, considering project's needs and opportunities for each student to exercise different professional practices. The project manager is ultimately responsible for planning and executing project's tasks but has the support of SEP professors, who form the management team, to plan and monitor the execution of students' tasks in the project.

As the duration of a project usually does not coincide with the academic semester term in which a SEP course section is taught, students can join an in progress project. In large projects, students may participate in only some iterations. Students can also participate in various projects to complement their practice hours throughout an academic semester. Furthermore, several projects may be carried out concurrently at SF so that students can participate in several projects and do not necessarily participate in the same projects.

3.3 SEP General Rules

The duration of the SEP course is one academic semester (16 weeks), and at least one SEP course instance is offered in every academic semester. The total workload of each course is 320 hours, with 20 hours per week, and takes place exclusively in the context of SF, which intends to integrate the practical activities developed by students (supported by professors) into the routine of projects developed at SF. This implies professors' direct and effective participation

[1] https://fabrica.inf.ufg.br/

in the planning, controlling, and evaluating actions developed in SF projects where SEP students participate.

SEP has all other courses from the SE undergraduate curriculum as prerequisites; that is, only after completing all the courses from the SE curriculum will students be able to enroll in a SEP course. This constraint ensures that students know how to execute all tasks demanded by the project since these tasks may involve practices in all kinds of SE activities within projects carried out at SF.

As SF does not have a dedicated physical space capable of accommodating the number of students and professors planned for a SEP course, they use the space of a university's computing laboratories and a meeting room (which is a conventional classroom reserved for the SEP course) to carry out their activities in SF projects. The maximum number of students in each class is limited by the classroom and laboratory capacities so that all students can work in the same space. Each professor of a SEP course is a preceptor to up to eight students. The few students per preceptor are essential due to the peculiarities of teaching activities carried out in the SF projects, which differ profoundly from those in the traditional courses.

In summary, all these premises and restrictions aim to ensure adequate conditions so that the software projects in which students exercise their skills are realistic in the sense of presenting the typical characteristics of software industry projects: the same set of people working in the same physical (or virtual) space, during the same hours, on consecutive days throughout each week, without context switching and with full-time management assistance.

4 EXPERIENCES IN CONDUCTING SEP COURSE INSTANCES

The execution of a SEP course instance follows a standardized procedure consisting of three stages for an academic semester: Preparation, Enactment, and Closure, as seen in Figure 1. These stages follow the Action Research cycle [45]: problem identification, solution planning, implementation, monitoring, and evaluation of effectiveness, leading to reflection and adjustment. Given our research question – *what approaches to undergraduate courses would provide adequate conditions for students to learn SE practice?* –, an initial planning was carried out in the Preparation Stage of each course semester; during the Enactment Stage, the implementation and monitoring were carried out, and in the Closure Stage the results were evaluated and identified improvements were applied in the next course instance.

The semester preparation stage begins approximately six weeks before the first class day. Professors and students who will participate in the course instance are identified, with the latter being allocated to projects conducted by the SF.

A project to be carried out during the semester is selected based on the number of enrolled students, considering the individual characteristics of the project, such as scope, complexity, development stage, and interdependencies between project activities. The selection of a project also considers the professors' competencies and the SF staff available, the project's maturity in terms of scope definition and requirements stability, and the availability of external stakeholders. The next step identifies SE knowledge areas involved in the project for the formation of the respective teams. For instance, a project in its initial phase may require a team focused on

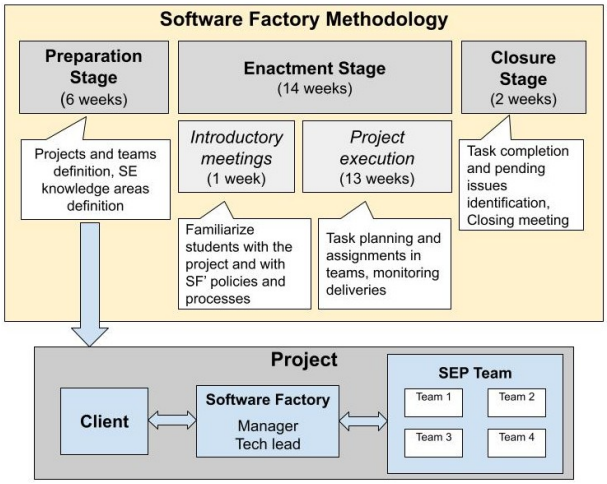

Figure 1: SEP course methodology procedure

requirements and user experience, while an ongoing project might demand implementation and testing teams. A series of meetings between SF collaborators and the professors begin, aiming to familiarize the latter with the SF processes and to contextualize them on the projects to be executed. In this pre-semester stage, students fill out a form to identify their profile. This allows a more appropriate allocation of students to specific project teams. Table 1 illustrates sample questions from the profile assessment form. Finally, an initial task backlog is created for each project team, covering at least two project iterations (sprints) that range from one to two weeks.

The Semester enactment stage begins with two introductory meetings. The first aims on integration, allowing everyone to introduce themselves and get familiar with one another. It is held during the first class with all SEP course participants: students, professors, and the SF team. Important information is provided in this meeting, such as the official communication channel for the project, and a Non-Disclosure Agreement (NDA) is signed when the project involves confidential data and intellectual property issues. In the second meeting, the target project and the composition of the teams are presented and discussed. In the remaining of the first week, the focus is on familiarizing everyone with the SF's policies, processes, and technologies.

From the second week on, the project is executed following an iterative process based on tasks controlled with the Git Flow branching model: i) team leader allocates task to student; ii) student creates local work branch; iii) student builds/changes artifacts; iv) student opens a pull request; v) SF team member inspects the artifacts, provides feedback and may request modifications; vi) student iteratively addresses the issues until the artifact is approved; vii) upon approval, the work branch is merged into the project's main branch (develop).

Throughout the semester, task deliveries are monitored to ensure the quality of the work results and adherence to project requirements. Examples of assigned tasks are: to design the context layer of the software architecture; to build the front-end code, or the test case, for a given use case; and to define an inspection checklist for use cases. Teams may be reorganized according to

Table 1: Examples of questions from the student profile assessment form

Questions	Response options
On Java environment, mark the concepts you are familiar with	JPA/Hibernate - SpringBoot - SpringData - Maven/Gradle - None - Others
On Angular framework, mark the concepts you are familiar with	Javascript/Typescript - CSS/HTML - RXJS - NPM - None - Others
On User Interface, mark the concepts you are familiar with	UX/UI - Style Guide - Design - Figma - Material Design - None - Others
On Software Quality, mark the concepts you are familiar with	Cypress - Test Cases - Functional testing - Quality metrics - None - Others
On Software Design, mark the concepts you are familiar with	UML - Design Patterns - PlantUML - C4Model - None - Others
In which SE areas do you have the most experience?	Requirements - Architecture - Construction - Testing - Inspection - None
Tell us about yourself, your experiences, and expectations	Open-ended

the project's evolution and needs and considering the impacts that teams (re)organization may have in SE projects [29].

The semester closure stage involves the last two weeks of a SEP course and is dedicated to completing ongoing tasks and identifying pending ones. Since the SF projects follow their schedules and are not restricted to the academic calendar, they continue even after the SEP course semester ends. On the last class day of a SEP course instance, a closing meeting is held where all deliverables produced during the SEP course semester are reviewed, along with lessons learned throughout a retrospective of the executed SE processes. So far, three SEP course instances have been completed. The following sections report their specific experiences.

4.1 SEP course instances

The inaugural SEP course instance occurred from July to November 2021. Due to the COVID-19 pandemic, activities were conducted remotely, with weekly meetings of the whole team aimed at monitoring and discussing tasks. The selected project had yet to begin and held social significance for brazilian educational policy[2]. An overview of the software solution was defined, identifying the technologies and components required for its execution and an initial backlog was established. Completing these tasks played a crucial role in defining the initial capability of the team and validating the knowledge gathered during profile collection. Initially, the students were introduced to the project, having the opportunity to contribute and participate in discussions about the definitions made by SF members and professors. This interaction proved to be important in the students training process since professionals in the early stages of their careers rarely have the chance to get involved in activities of this nature. Three essential components of the project were outlined and assigned to specific teams: a team (3 students) was tasked with developing the dashboard component; a second team (2 students) with developing the management component; and, lastly, a data science team (3 students) was designated to provide the necessary data for the other two components. One student created the system's visual identity and user interface design.

The second SEP course instance occurred from May to September 2022, and all activities were carried out in a distance learning format due to the COVID-19 pandemic. The students developed a new component to integrate into software that had been created by the students of the SEP course in the previous semester. Three SE knowledge areas were applied: software implementation and integration (8 students), software requirements (4 students), and software testing (1 student).

The third SEP course instance occurred from April to August 2023. This course instance conducted its activities in a hybrid format, i.e., with remote and in-person activities. Beyond the projects' specific activities, students could engage in other SF's SE concerns, including definition, documentation, or improvement of SE processes. There were four teams: development, to finish a component initiated in the first SEP course instance (3 students); quality; to improve and document the Verification and Validation (V&V) SF processes (4 students); test, focused on the evolution of the test automation process (3 students); and architecture, to define and document a new SF software architecture process (3 students).

Several typical SE project issues appeared in each SEP course instance, such as difficulties with workstations' configuration, lack of precision of some requirements, or use of different criteria for monitoring students' soft skills. Our experience indicates that the issues that appear in each SEP course instance contributes to the experience of the students, helping them to learn how to solve problems in an uncontrolled environment.

5 SEP AND SE COMPETENCIES

The SEP course aims to offer students a software development experience in an environment close to the reality of the SE industry. Thus, in addition to technical and technological (knowledge and cognitive) skills, students need to experience their emotional competencies (emotional skills and attitudes) in building software in a team for an external client. The latest instance of the SEP course aimed to honor not only the experience with technical and technological skills but also to offer the opportunity to put into practice emotional and cognitive skills in teamwork. This experience showed a valuable interaction between professors and students. The professors played the role of *preceptors*, a term taken from the health education area and used in this course to reference the professor's responsibility for managing and guiding students in their professional activities. The students were divided into groups of three to four components and allocated to one SE knowledge area: Architecture/Design, Construction, Testing, and Verification and Validation (V&V). Each preceptor was mentoring and assessing students in one technical knowledge area, and all professors observed the students' ethical attitudes and professional postures [5]. The cognitive skills practiced and assessed in the project were reasoning, analytical skills, problem-solving, and innovation. The emotional abilities practiced were aptitude, initiative, enthusiasm,

[2]https://tceduca.irbcontas.org.br/mapa

work ethic, willingness, trustworthiness, team participation, technical leadership, and communication skills. These also included effort, punctuality, attendance, collaboration, and respect. The third instance of the course was carried out in the context of a competency management process, comprising the activities of planning, monitoring, evaluation and feedback, and competency development.

5.1 Planning and monitoring competencies

The preceptors organized the course based on project's needs and student profiles. The knowledge areas to be worked on were established, using the SWEBOK [25] as a reference. The students were allocated to process areas with activities and technological tools to support the process. The process was reviewed, improved, and expanded weekly according to the student's performance. The preceptors defined final products for each team rather than intermediate deliverables, as this would depend on the practices to be carried out by the students. The respective products/deliverables were incorporated into the team results as the process was executed. While still planning, the tutors defined assessment criteria and respective marks/weights for competencies that should be developed.

The evaluation criteria covered the quality of the products and the expected results, compliance with defined processes and standards, and demonstration of developed competencies, cognitive skills, technical skills, and emotional skills, as well as the attitude represented by the student's professional behavior and posture throughout the construction of the product. Seven monitoring cycles were defined for the activities, tasks, and products. In the SEP course, the two perspectives of competency - American and French schools - are combined, involving the capacity that the student must demonstrate throughout the execution of the project and the respective results and products. Table 2 shows the minimum competencies that should be developed. The execution process developed by the students was improved and expanded within each monitoring and evaluation (feedback) cycle.

Monitoring was carried out weekly by the preceptors with all the teams. Each preceptor was responsible for the class on a specific day of the week. The preceptors monitored the students' activities during the class period, including the technical activities of the area in which they were responsible. At this stage, the preceptor clarified doubts and provided guidance and suggestions regarding activities and products developed during the period. Technical competencies were monitored according to the preceptor's expertise, and all preceptors analyzed cognitive and emotional competencies according to the outcomes of the student work, defined in Table 2.

5.2 Evaluation and Feedback of competencies

Initially, evaluation and feedback activities were carried out weekly, along with monitoring, focusing on competencies development [16]. However, after the second week, it emerged that a more extended timeframe was necessary for the students to implement corrective and improvement actions and suggestions associated with personal performance, the SE process, and the delivered products. Since then, the preceptors have carried out feedback every fortnight, according to the assessment criteria defined in the semester preparation stage, involving analysis of the products built and the development of technical, emotional, and cognitive skills [19].

The students received feedback on their work through both the advising of professors and the SF review process. Students were assigned supervisors who provide guidance, feedback, and help them understand best practices for the competency of interest. Experienced SF team members reviewed student's artifacts, offering technical feedback and hints for potential improvements. Regular team meetings provided opportunities for students to present their progress to their supervisor and teammates, discuss challenges, and receive feedback. Each student evaluated his/her own improvement and his/her teammates' evolution using a 360º survey. The professors shared a spreadsheet to register grades for each review and to monitor progress in soft skills, using the same criteria. Besides quantitative results, the shared spreadsheet records qualitative impressions of each professor in relation to the evolution of each student. In the term of the course, a survey was conducted, as seen in Table 3, and the students gave feedback of their experience on the course. In all three instances, the students' opinion was of great satisfaction. The course instruments and data are available in a SF repository[3].

Each preceptor had their evaluation style. Some did the assessment and feedback only as a group; others did group and individual assessments and provided both group and personal feedback. The evaluations and feedback provided a lot of reflection for most students and can be seen in the behavior change. There were four categories of assessments:

- carried out by the preceptor for each student;
- carried out by the preceptor for the team as a whole;
- self-assessment by the students themselves;
- 360º assessment, in which the other team members assessed a team member.

The competencies assessed in each feedback cycle, described in Table 2, were inspired by, but not limited to, the SWECOM SE competencies and skills [26].

5.3 Development of competencies

As a result of the assessments and feedback provided by the preceptors, the students in groups and individually improved their attitudes and practices throughout the project. This can be seen in the performance of product quality, attendance, and punctuality of delivery of intermediate and final products. In all, each type of evaluation was carried out six times by the preceptors, teammates, and the student himself (self-assessment). This development was evaluated regarding the observed behavior, the lessons learned from the preceptors' guidance and feedback, and the opportunities for improvement that the students identified.

Figure 2 shows the students' progress in developing technical skills. The average results of the first assessment are shown in blue and the last in orange. There was a significant improvement in competencies between the first and last assessment cycles, and several factors have contributed to this result. The first factor is students' low seriousness towards SEP professional attitudes requirements at the beginning of the course. Most students considered SEP as a traditional course and, initially, they needed to meet attendance and punctuality requirements. The initial evaluations clarified to the students that this course enrollment was similar to working

[3]https://fabricadesoftwareinf.github.io/sbes2024/

Knowledge	Cognitive Skill	Personal Emotional Skill	Social Emotional Skill	Outcomes
allocated SE knowledge area practices	comprehension, analysis, communication	initiative, enthusiasm, availability, ethical behavior	communication, teamwork, leadership	results of activities in the knowledge area
SE tools	comprehension, analysis, communication	aptitude, initiative, availability, ethical behavior	communication, teamwork, leadership, respect	adequate use of tools
process activities, and respective products	abstraction, application, reasoning, synthesis	initiative, availability, ethical behavior	communication, teamwork, leadership, respect	process definition and compliance

Table 2: Minimum SEP Competencies

Table 3: Final evaluation questionnaire

#	Questions	Response type
1	The course contributed to your education (i.e., learning)	Likert Scale: Agree - Disagree
2	The course has helped you to practice techniques learned in other undergraduate courses	Likert Scale: Agree - Disagree
3	You were familiar with methods, techniques, and tools needed for working in the project	Likert Scale: Agree - Disagree
4	Working in a project team contributed to developing your teamwork skills	Likert Scale: Agree - Disagree
5	The team organization is the best work format for a software development project	Likert Scale: Agree - Disagree
6	The course has helped you to better understand the challenges of software development	Likert Scale: Agree - Disagree
7	The course educational methodology helped you in your tasks in the project development	Likert Scale: Agree - Disagree
8	The communication between students and instructors was improved by the course methodology	Likert Scale: Agree - Disagree
9	What were the main challenges you faced on working in the project?	Open-ended
10	Would you recommend this course, as it was held in this semester, to a classmate? Why?	Yes or No – Open-ended
11	Final Comments: use this space for additional comments concerning the course.	Open-ended

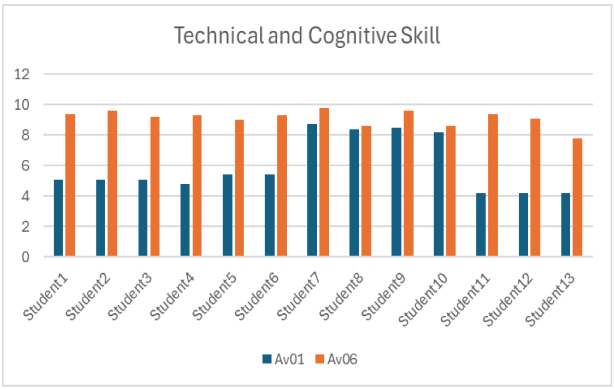

Figure 2: Technical competencies evolution

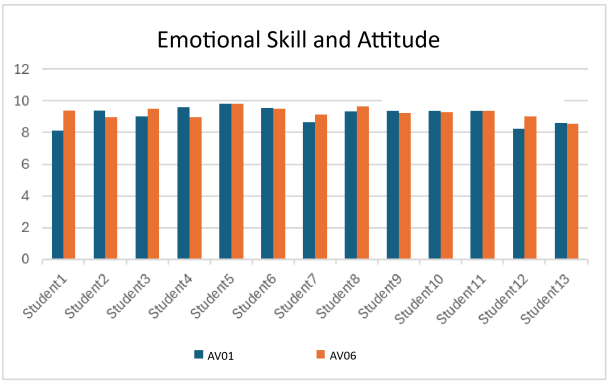

Figure 3: Emotional competencies evolution

in a company and that a professional attitude was mandatory. Another critical factor was that, as the monitoring, evaluations, and feedback took place, students realized that the team perceived the weaknesses in their work, so they tried to correct mistakes, and the level of cooperation increased.

The emotional skills and attitudes were divided into individual and social. Figure 3 shows average results of the first (in blue) and last (in orange) assessments. The little change in these values may be related to multiple influencing factors on emotional skills, such as the person's personality, the environment in which they live, their values, and beliefs. These aspects are challenging to change and nevertheless strongly influence the application and exercise of technical skills. Another factor that justify this result is the difficulty in perceiving changes in personal attitudes and soft skills.

6 RESULTS AND LESSONS LEARNED

This section synthesizes the main results of the experiences conducting the SEP course in three academic semesters.

6.1 Role of the Professor in SEP

The circumstances in which activities are carried out in SE projects are different from those that occur in isolated courses or academic work. Therefore, SEP professors must primarily play the role of

preceptor for their students. The preceptor is responsible for theory-practice integration throughout the project, teaching, supervising, guiding, and leading students in the effective practice of their future profession. To act as a preceptor, the professor must have a small number of students so that they can satisfactorily guide the practical activities essential to the training of a Software Engineer.

A limited number of students per professor ensures that the professional acts performed by students in SEP have measured quality, with planning and constant monitoring of students' actions, guaranteeing the security of information and contributing to the formation of a well-prepared and capable professional, with appropriately structured skills to act as a SE professional, valuable and reliable to society. As a SEP course has 320 hours per semester, each professor may have a workload of up to 20 hours per week (320 hours / 16 weeks). Therefore, the professor should be dedicated exclusively to this course, as in addition to being a preceptor for a group of students, the professor is a member of the management team of the projects in which SEP is involved. The professors must focus on SEP's projects to contribute to project management and effectively monitor all their students' activities.

6.2 Students' Performance Assessment Method

SEP is based on formative assessment and emphasizes monitoring students throughout the academic period to verify progress toward the proposed objectives, using a humanistic approach [44]. Through formative assessment, students become aware of their mistakes and successes and find encouragement to continue their studies systematically. This formative assessment must be a quality control instrument, including self-evaluation, peer evaluation within projects, and evaluation by the course's professors. This assessment form aims to complement learning, guide students' actions throughout the process, and avoid tensions caused by traditional assessments. Based on the educator's sensitivity and technical perspective, this assessment format provides more information that allows the teaching process to be adapted to each person's needs. To achieve this, the professor must monitor a small number of students so that the focus of assessment and training is on the individual.

Individual assessments of skills and attitudes, based on general checklists on ethics and professional attitude, must be carried out every two weeks by all professors of a SEP course instance on all enrolled students. These assessments consider the student's self-assessment and the assessment made by project peers. Furthermore, professors carry out a more specific and in-depth evaluation of the individual competency of their students, observing the proper application of specific knowledge and the contribution to the project objectives. The student's final grade is a weighted average of behavioral (social and personal) and professional (managerial, technical, and technological) assessments. The behavioral competencies contribute to the final grade with a weight of four, and the professional competencies with a weight of six.

6.3 SEP as a University Outreach Activity

University Outreach in Brazilian higher education is a set of educational activities that promotes a transformative interaction between higher education institutions and other sectors of society for production and application of knowledge in articulation with teaching and research. The Brazilian Ministry of Education requires that university outreach activities accomplish ten percent of the total workload of undergraduate education programs. In the SE bachelor's degree program in which the SEP course is inserted, the whole university outreach workload is accomplished within this course. The SF project portfolio management imposes mandatory interaction with external entities, and projects that involve SEP students are exclusive to non-profit organizations unrelated to the university structure. These organizations seek support from SF in the face of difficulties afflicting them, which can be remedied by applying SE to create software products and services. The SF's client organizations expect that the project results meet their needs and are willing to expose their problems and difficulties so that, in addition to acts inherent to SE, pedagogical actions may also be executed in the form of practical analysis activities carried out by students under the direct supervision of professors. This scenario allows the professor to simultaneously perform didactic and university outreach actions, which must be well understood both by those interested in the project and by the professors and students involved so that the ethical principles of SE are strictly observed and followed. To achieve this, each professor must monitor a small number of students to ensure that:

- professional acts carried out by students have their quality assessed, aiming to satisfy project stakeholders' objectives;
- pedagogical actions are practical, with planning and constant monitoring of students' actions within the project;
- information security and protection of users and project sponsors' rights are preserved.

Once these conditions are guaranteed, SEP becomes ideal for *university outreach* activities. By participating in projects with external stakeholders and working in various types of SE activities, students face highly complex professional situations within the project, improving their use of SE principles and technologies without losing sight of the fact that the object of their attention must be the satisfaction of the needs of those interested in the project who, in turn, represent the community in which the SF operates. Therefore, the students' activities include aspects of learning that benefit the students and aspects of university outreach that benefit external non-profit organizations. This strengthens the desired inseparability between learning and outreach activities, which are two fundamental academic objectives.

6.4 Benefits for SE Student Education

For the competencies maturation to be effective, students must experience different SE processes that are learned in several courses within the SE bachelor's Degree curriculum and can be organized into technical and management processes. Both are supported by numerous SE tools and mastering these tools' technologies is a mandatory SE competency. Thus, SE professionals must be able to carry out activities involving technical, managerial, and technological competencies. Real SE projects, like those at SF, typically exercise all three competencies. It is essential to contrast this realistic project scenario, where most types of competencies are exercised, with the traditional courses practical activities scenario, where the focus is directed to just one kind of activity in a controlled environment.

Another advantage of the practical activities taught in SEP is that SF projects aim to meet needs of real users and sponsors; that is, projects not only have academic and pedagogical interests but are mainly aimed at generating products and provisioning effective SE services for the external community. In this way, projects meet quality requirements defined by (a) project's stakeholders (users and sponsors), (b) applicable SE technical standards and best practices (which are professors' concerns), and (c) organizational SF policies and processes. Furthermore, projects consider cost, deadline, and scope restrictions negotiated with those interested parties, providing students with an experience of the difficulties of reconciling different expectations involved in a software project. The insertion of the SEP course in SF projects also allows students to interact with users, sponsors, and SE professionals who work at SF, dealing with real problems and assuming increasing responsibilities as relevant and active agents in their community.

Close monitoring by a professor with a reduced group of students contributes to the quality of these students' training, ensures the quality of results produced by these students for the project in which they work, and guarantees the necessary confidentiality in real SE projects, since a restricted group of students, under the supervision of a professor, will have access to confidential information of each project. For this reason, professors and students sign a confidentiality agreement with SF before starting their project activities, reinforcing concepts of ethics in SE.

6.5 Benefits for Software Factory

The inclusion of students in the SF team not only enhances its productivity but also reduces the costs associated with software development. Additionally, when funding is available, these resources are directed toward stimulating innovation and promoting human resource development within the educational institution. This includes providing scholarships for undergraduate students, which helps combat attrition in higher education courses.

Students can also bring new ideas and perspectives to projects, fostering innovation and creativity within the Software Factory. Moreover, by actively engaging in training professionals prepared for the job market, the SF strengthens its reputation, attracting more partnerships and investments from the industry. Supervising and mentoring students with diverse backgrounds and skills allows the SF team to develop its management and leadership capabilities. This brings long-term benefits, particularly in fostering a culture of continuous professional development.

7 CONCLUSIONS

This paper reports results and lessons learned from applying an innovative approach to teaching SE practice in an undergraduate course. The essence of our approach is practical learning in an academic SF that represents for SE students what a university hospital represents for medical students. There are other ways to learn SE practice, but we believe that academic software factory-based learning is the most effective. We hope that this article motivates universities to create their own academic software factories, as our experience shows that the investment is relatively small compared to the benefits generated for students and society as a whole. Including students in a real software production environment, detached form a purely academic context, is the cornerstone of the proposed educational process. It promotes students' professional attitudes and competencies and leads students to consolidate learning through doing and be active subjects in their learning process.

The SEP course aims to promote SE students' competencies maturity and develop their technical, social, and personal skills and attitudes as future competent SE professionals and citizens committed to their community and the society they belong to. This maturation takes place in the final course of the SE Bachelor's Degree curriculum, reinforcing the practice of soft and hard skills and consolidating knowledge students acquired throughout the theoretical-practical coursework in an environment that realistically represents situations that will be experienced in their professional career. In the SEP coursework, students participate in complex activities, maintain commitments, and respond to absolute obligations. It is an opportunity for students to be evaluated regarding their ethical and professional attitude, respect for institutional standards, and relationship with project stakeholders, including users, sponsors, and SF's staff. The course concept favors active practice-based learning processes, focusing on ethics and attitudes of the future SE professional towards challenges that occur when working within a software-producing organization. This experience is essential to their future professional life since facing real difficulties during training improves students' understanding and commitment to the society in which they operate. It stimulates and values ethical and humanistic dimensions in the training, developing attitudes, principles, and values of citizenship oriented to the society development.

SF projects are developed to serve the community and involve multi-professional and interdisciplinary tasks that integrate different competencies of SE. SEP course operates in these SF projects that develop integrated activities between university and its community, favoring practical learning based on scientific methodology and integrated into the university service provision system. SF projects involving SEP course also serves as a continuing education strategy for SF staff, functioning as centers that generate resources, produce knowledge, and bring together highly qualified human resources. This approach can contribute to provide adequately trained professionals, since this is one of the biggest obstacles to the growth of the software industry in Brazil and around the world.

REFERENCES

[1] ACM and IEEE. 2016. Software engineering Code of Ethics. https://www.computer.org/education/code-of-ethics.

[2] ACM and IEEE. 2020. Software Engineering Curriculum. Guidelines for Undergraduate Degree Programs in Software Engineering. https://www.acm.org/education/curricula-recommendations.

[3] Deniz Akdur. 2022. Analysis of Software Engineering Skills Gap in the Industry. *ACM Trans. Comput. Educ.* 23, 1, Article 16 (dec 2022), 28 pages. https://doi.org/10.1145/3567837

[4] André Antunes, Daltro Nunes, Jair Leite, and Marcelo Yamaguti. 2017. *Bacharelado em Engenharia de Software.* Sociedade Brasileira de Computação (SBC), Porto Alegre, RS, Brazil, Chapter IV, 56–78.

[5] Renata Araújo, Alcides Calsavara, Alessandro Cerqueira, and Jair Leite. 2019. *Referenciais de Formação para os Cursos de Graduação em Computação no Brasil - Competências Atitudinais.* Technical Report. Sociedade Brasileira de Computação (SBC). 11 pages.

[6] Nana Assyne, Hadi Ghanbari, and Mirja Pulkkinen. 2022. The state of research on software engineering competencies: A systematic mapping study. *Journal of Systems and Software* 185 (March 2022), 111183.

[7] Mirza Zaeem Baig, Muhammad Usman Ul Haq, Hafiz Muhammad Umer Surkhail, Rabika Iqbal, and Muhammad Mohsin Sheikh. 2018. Bridging the industry-academia collaboration gap a focus towards final year projects. In *Proceedings of the 2nd International Conference on High Performance Compilation, Computing and Communications* (Hong Kong, Hong Kong) *(HP3C)*. Association for Computing Machinery, New York, NY, USA, 40–44. https://doi.org/10.1145/3195612.3195620

[8] Renata Brasil-Silva and Fábio Levy Siqueira. 2022. Metrics to quantify software developer experience: a systematic mapping. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. ACM, Virtual Event, 1562––1569. https://doi.org/10.1145/3477314.3507304

[9] Manuel Caeiro-Rodríguez, Mario Manso-Vázquez, Fernando A. Mikic-Fonte, Martín Llamas-Nistal, Manuel J. Fernández-Iglesias, Hariklia Tsalapatas, Olivier Heidmann, Carlos Vaz De Carvalho, Triinu Jesmin, Jaanus Terasmaa, and Lene Tolstrup Sørensen. 2021. Teaching Soft Skills in Engineering Education: An European Perspective. *IEEE Access* 9 (2021), 29222–29242. https://doi.org/10.1109/ACCESS.2021.3059516

[10] Jonathan Cazalas, Christian Roberson, and Zeeshan Furqan. 2024. From Degree to Developer: the Creation and Evolution of a CS Course Designed to Bridge the Academia-Industry Gap. In *Technical Symposium on Computer Science Education (SIGCSE)*. ACM, Portland, OR, USA, 186–192. https://doi.org/10.1145/3626252.3630860

[11] 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.

[12] Marcela da C. de Souza, Sandro R. Oliveira, and Sílvio R. Meira. 2017. A Systematic Review to Assist in Identifying Teaching Approaches to Guide the Application of an Interdisciplinary Software Factory in IT Undergraduation. In *Simpósio Brasileiro de Engenharia de Software*. SBC, Fortaleza, CE, Brazil, 384—-391.

[13] Ray Dawson and Ron Newsham. 1997. Introducing software engineers to the real world. *IEEE Software* 14, 6 (1997), 37–43. https://doi.org/10.1109/52.636640

[14] Birgit Demuth, M. Fischer, and Heinrich Hussmann. 2002. Experience in early and late software engineering project courses. In *Proceedings 15th Conference on Software Engineering Education and Training (ICSE-SEET)*. IEEE, Kentucky, USA, 241–248. https://doi.org/10.1109/CSEE.2002.995216

[15] Anastasia Dimitriadou, Naza Djafarova, Ozgur Turetken, Margaret Verkuyl, and Alexander Ferworn. 2021. Challenges in serious game design and development: Educators' experiences. *Simulation & Gaming* 52, 2 (2021), 132–152.

[16] Joelle Ducrot and Venky Shankararaman. 2015. Measuring student performance and providing feedback using Competency Framework. In *International Conference on Engineering Education - ICEED*. IEEE, Kanazawa, Japan, 55–60. https://doi.org/10.1109/ICEED.2014.7194688

[17] Thomas Durand. 2015. L'alchimie de la compétence. *Revue Francaise de Gestion* 253, 8 (2015), 267–295. https://doi.org/10.3166/RFG.160.261-292

[18] Sigrid Eldh and Sasikumar Punnekkat. 2012. Synergizing industrial needs and academic research for better software education. In *International Workshop on Software Engineering Education Based on Real-World Experiences, EduRex*. IEEE, Zurich, Switzerland, 33–36. https://doi.org/10.1109/EduRex.2012.6225703

[19] Reza Fauzan, Daniel Siahaan, Mirotus Solekhah, Vriza Wahyu Saputra, Aditya Eka Bagaskara, and Muhammad Ihsan Karimi. 2023. A Systematic Literature Review of Student Assessment Framework in Software Engineering Courses. *Journal of Information Systems Engineering and Business Intelligence* 9, 2 (2023), 264–275. https://doi.org/10.20473/jisebi.9.2.264-275

[20] Nuno Flores, Ana CR Paiva, and Nuno Cruz. 2020. Teaching software engineering topics through pedagogical game design patterns: An empirical study. *Information* 11, 3 (2020), 21.

[21] Ivan Garcia, Carla Pacheco, Andrés León, and Jose A Calvo-Manzano. 2020. A serious game for teaching the fundamentals of ISO/IEC/IEEE 29148 systems and software engineering–Lifecycle processes–Requirements engineering at undergraduate level. *Computer Standards & Interfaces* 67 (2020), 103377.

[22] Vahid Garousi, Gorkem Giray, Eray Tuzun, Cagatay Catal, and Michael Felderer. 2019. Aligning software engineering education with industrial needs: A meta-analysis. *Journal of Systems and Software* 156 (Oct. 2019), 65–83.

[23] Eiji Hayashiguchi, Hironori Washizaki, Katsutoshi Shintani, and Daisuke Yoshioka. 2022. The Competency-based Computing Curricula 2020 and SFIA V7 comparison focusing on Digital Transformation Age. In *World Engineering Education Conf.* IEEE, Santos, Brazil, 1–6. Issue V.

[24] Nicole Herbert. 2018. Reflections on 17 Years of ICT Capstone Project Coordination: Effective Strategies for Managing Clients, Teams and Assessment. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 215–220. https://doi.org/10.1145/3159450.3159584

[25] IEEE Computer Society. 2014. *Guide to the Software Engineering body of knowledge (SWEBOK) V3*. IEEE Computer Society Press, Washington, DC, United States. https://ieeecs-media.computer.org/media/education/swebok/swebok-v3.pdf

[26] IEEE Computer Society. 2014. *Software Engineering Competency Model) V1*. IEEE Computer Society Press, Washington, DC, United States. https://https://www.computer.org/volunteering/boards-and-committees/professional-educational-activities/software-engineering-competency-model

[27] ISO, IEC, and IEEE. 2017. *International Standard 12207 - Systems and software engineering — Software life cycle processes*. ISO/IEC/IEEE, Geneva, CH.

[28] Letizia Jaccheri and Patricia Lago. 1998. How project-based courses face the challenge of educating software engineers. In *Proc. of the joint World Multiconference on Systemics, Cybernetics and Informatics (SCI'98) and the 4th International Conference on Information Systems Analysis and Synthesis (ISAS'98), Lecture Notes in Computer Science*, Vol. 750. International Institute of Informatics and Systemics, Orlando, USA, 377–385.

[29] Julio Juárez, Cipriano Santos, and Carlos Brizuela. 2021. A comprehensive review and a taxonomy proposal of team formation problems. *Comput. Surveys* 54 (2021), 153–186.

[30] Ze Shi Li, Nowshin Nawar Arony, Kezia Devathasan, and Daniela Damian. 2023. "Software is the easy part of Software Engineering" - Lessons and Experiences from A Large-Scale, Multi-Team Capstone Course. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. ACM/IEEE, Melbourne, Australia, 223–234. https://doi.org/10.1109/ICSE-SEET58685.2023.00027

[31] Georgios Liargkovas, Angeliki Papadopoulou, Zoe Kotti, and Diomidis Spinellis. 2022. Software Engineering Education Knowledge Versus Industrial Needs. *IEEE Transactions on Education* 65, 3 (2022), 419–427. https://doi.org/10.1109/TE.2021.3123889

[32] Stephanie Ludi and James Collofello. 2001. An analysis of the gap between the knowledge and skills learned in academic software engineering course projects and those required in real: projects. In *31st Annual Frontiers in Education Conference. Impact on Engineering and Science Education*. IEEE, Reno, NV, USA, T2D–8. https://doi.org/10.1109/FIE.2001.963881

[33] Joseph Maguire, Steve Draper, and Quintin Cutts. 2019. What Do We Do When We Teach Software Engineering?. In *Proceedings of the 2019 Conference on United Kingdom & Ireland Computing Education Research*. ACM, Canterbury United Kingdom, 1–7.

[34] Maíra R. Marques, Alcides Quispe, and Sergio F. Ochoa. 2014. A systematic mapping study on practical approaches to teaching software engineering. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. IEEE, Madrid, Spain, 1–8. https://doi.org/10.1109/FIE.2014.7044277

[35] José Metrôlho, Fernando Ribeiro, Paula Graça, Ana Mourato, David Figueiredo, and Hugo Vilarinho. 2022. Aligning software engineering teaching strategies and practices with industrial needs. *Computation* 10, 8 (2022), 129.

[36] Lennart Ohlsson and Conny Johansson. 1995. A practice driven approach to software engineering education. *IEEE Transactions on Education* 38, 3 (1995), 291–295. https://doi.org/10.1109/13.406508

[37] Kai Qian, Dan Lo, Reza Parizi, Fan Wu, Emmanuel Agu, and Bei-Tseng Chu. 2018. Authentic Learning Secure Software Development (SSD) in Computing Education. In *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, San Jose, California, USA, 1–9. https://doi.org/10.1109/FIE.2018.8659217

[38] Eric Ras, Ralf Carbon, BjÖrn Decker, and JÖrg Rech. 2007. Experience Management Wikis for Reflective Practice in Software Capstone Projects. *IEEE Transactions on Education* 50, 4 (2007), 312–320. https://doi.org/10.1109/TE.2007.904580

[39] Mira Kania Sabariah, Veronikha Effendy, Jati H. Husen, Daffa Hilmy Fadhlurrohman, and Rony Setyawansyah. 2023. Experiences With Gap-Bridging Software Engineering Industry-Academia Collaborative Education Program. In *2023 IEEE 35th International Conference on Software Engineering Education and Training (ICSE-SEET)*. IEEE, Tokyo, Japan, 168–172. https://doi.org/10.1109/CSEET58097.2023.00035

[40] Mihaela Sabin, John Impagliazzo, Hala Alrumaih, Cara Tang, and Ming Zhang. 2018. IT2017 report: Implementing a competency-based information technology program. *SIGCSE 2018 - Proceedings of the 49th ACM Technical Symposium on Computer Science Education* 2018-Janua (2018), 1045–1046. https://doi.org/10.1145/3159450.3159636

[41] Javier Saldaña-Ramos, Ana Sanz-Esteban, Javier García, and Antonio Amescua. 2014. Skills and abilities for working in a global software development team: a competence model. *Journal of Software: Evolution and Process* 26, 3 (Feb. 2014), 329–338. https://doi.org/10.1002/smr.1588

[42] Venky Shankararaman, Paul M. Leidig, Greg Anderson, and Mark Thouin. 2021. IS2020:Competency-Based Information Systems Curriculum Guidelines. *Proceedings - Frontiers in Education Conference, FIE* 2021-Octob (2021), 1–4. https://doi.org/10.1109/FIE49875.2021.9637150

[43] Simone S. R. Souza, Bruno H. Oliveira, Filipe Grillo, and Christian de Cico. 2016. Construção de Plataformas Digitais durante o Ensino de Engenharia de Software: um relato de Experiência. In *Anais do IX Fórum de Educação em Engenharia de Software*. SBC, Maringá, Brasil, 13–22.

[44] Marcus Vinicius Alencar Terra, Vanessa Tavares De Oliveira Barros, and Rodolfo Miranda De Barros. 2022. Performance Management of IT Professionals: A Humanistic Model. In *Conference on Computer Science and Intelligence Systems, FedCSIS*. IEEE, Sofia, Bulgaria, 721–729. https://doi.org/10.15439/2022F220

[45] Michel Thiollent. 2005. Insertion of action-research in the context of continued university education. *International Journal of Action Research* 1, 1 (2005), 87–98.

[46] Veronika Thurner and Axel Böttcher. 2012. Expectations and deficiencies in soft skills. In *Global Engineering Education Conference (EDUCON)*. IEEE, Marrakech, Morocco, 1–7. https://doi.org/10.1109/EDUCON.2012.6201197

[47] Denis O. Zmeev and Oleg A. Zmeev. 2020. Project-Oriented Course of Software Engineering Based on Essence. In *Conference on Software Engineering Education and Training (ICSE-SEET)*. IEEE, Munich, Germany, 296–298. https://doi.org/10.1109/CSEET49119.2020.9206240