CLEI Electronic Journal, Volume 28, Issue 5, paper 6, September 2025

Active Learning Pedagogical Strategies in Software Engineering University Courses

Juan Ignacio Irabedra
Universidad ORT Uruguay,
Montevideo, Uruguay, 11200,
juanignacioirabedra@gmail.com

and

Martin Solari
Universidad ORT Uruguay,
Montevideo, Uruguay, 11200,
martin.solari@ort.edu.uy

and

Gaston Mousques
Universidad ORT Uruguay,
Montevideo, Uruguay, 11200,
mousques@ort.edu.uy

Abstract

Context: Software Engineering solves real world problems and thus requires future engineers to amass experience solving problems in authentic contexts during their formal education. Active learning provides learning contexts where students can acquire experience by doing contextualized and meaningful activities that involve high-order thinking. Despite active learning being proved to be more effective providing hands-on experience in software engineering situations, traditional learning activities such as lectures still rival the former.
Objectives: Identify and define what active learning strategies are used in software engineering higher education (1) and what application experiences of such strategies have been conducted between 2018 and 2023 (2).
Method: We conducted a Rapid Review to identify active learning strategies applied in Software Engineering in higher education.
Results: 18 active learning strategies on software engineering were found across 49 studies. Games and Project-Based Learning were the most frequently used strategies.
Conclusions: There are diverse active learning strategies used in software engineering higher education. Active learning strategies can be deployed in varied higher education settings and promote competencies development. Active learning strategies can be combined with other active learning strategies as well as combined with traditional learning activities like lectures in order to foster both lower and higher order thinking skills.

Keywords: Active learning, software engineering, higher education

1 Introduction

Software Engineering entails activities such as the development, operation and evolution of digital products in multidisciplinary teams under quality, time and costs constraints. The way in which such activities are conducted has evolved, and engineering is not limited to technical expertise anymore. Agile Software

Engineering [1] has diverged towards a discipline focused on adaptation rather than following plans. Collaboration and interpersonal interactions became keystone on management situations, and working software became a priority over exhaustive documents.

Software engineering professionals necessitate both technical knowledge as well as other skills [2] like cooperation, effective communication, self-learning, empathy among many others. These skills are often referred as soft skills and have become as important as technical knowledge within the field. Not only are soft skills desirable in software engineering but they have also become a formal requirements on job applications [3]. All these abilities have to be acquired and developed during higher education while simultaneously amassing hands-on experience on software engineering challenges.

Software engineering students often fail to meet industry expectations. It has been proven that deficiencies exist on technical skills such as configuration management and non technical skills like leadership and communication [4]. This gap between the software industry and universities leads to question how might universities adapt the way Software Engineering is currently taught. This reasoning aligns with the first research objective (1) of the current study.

Therefore, it is relevant to discuss what learning activities can foster both technical skills development and soft skills and hands-on experience simultaneously. The most commonly used teaching approach, lectures, can hardly provide them all. Students must do more than just listening in order to become ready to face software engineering challenges at the pace of the industrial sector.

Active learning involves strategies that allow individuals to amass experience on software engineering skills in authentic contexts. This learning approach offers more then traditional learning activities like passively listening [5] [6].

Active learning is often defined through examples. Defining active learning is a hard task and has seldom been theoretically defined. Prince defined active learning [7] as instructional methods that engage students in the learning process with meaningful activities and high-order thinking.

Active learning has shown to be effective in software engineering education [7] and appropriately complements traditional learning approaches [8].

This work is an extension of [9] and seeks to identify and define what active learning strategies are used in software engineering (1) as well as what application experiences there are for such strategies (2) in Software Engineering education between 2018 and 2023. The extension purpose is to report active learning strategies application across an entire Software Engineering Department at Universidad ORT Uruguay.

The study of the deployment of active learning strategies was conducted during an undergraduate capstone project which sought to systematize knowledge about pedagogical approaches within the Software Engineering Department.

Knowing what active learning strategies are available allows instructors to broaden their teaching tools and innovate within the classroom.

2 Related work

In 2014 a systematic mapping study on practical approaches for teaching software engineering was conducted [10]. This study considered 174 primary studies on applications of software engineering learning approaches between 1982 and 2013. One of the research objectives of such systematic mapping study was to identify what learning approaches can provide hands-on experience to software engineering students. Learn by doing was the most frequently found approach, present in 54% of the studies considered. Project-Based Learning was the second most frequently found approach, present in 21% of the studies. Case study and Games were found in 9% and 7% of the studies respectively. Other approaches were present in 5% or less of the studies, such as simulations, traditional teaching, flipped classes and Service Learning. The aforementioned mapping study answers a question identical to the first research objective (1) of this study, but spans a different time horizon.

Another systematic mapping study in 2023 aimed at identifying what learning approaches can be used for teaching requirements engineering [11]. Such study considers only a sub-discipline of software engineering. It arrives to the conclusion that practical, hands-on learning approaches have become a trend in higher education and that theory-based instruction plays a subordinate role in requirements engineering education.

In 2022 a secondary study was carried out on Project-Based Learning effectiveness [12] in software engineering education. Such secondary study reports 20 primary studies on Project-Based Learning application in software engineering education. The study concludes that Project-Based Learning is both effective for developing technical and non-technical skills.

Additionally, there is work reporting the application of specific activities carried out in Software Engineering courses such as [13].

3 Method

With the purpose of finding and defining active learning strategies that were applied in Software Engineering higher education (1) and what application experiences there are for such strategies (2) a Rapid Review (RR) was conducted. A Rapid Review is a secondary study that improves the velocity of a systematic mapping study of literature [14]. Rapid Reviews are valuable in contexts where results are expected to be delivered quickly in order to make decisions.

In this study an application experience is a situation where a particular active learning strategy has been applied within a single course or series of courses.

Rapid Reviews trade off precision for resources. Rapid Reviews are often shorter and can not span as much as full-scale systematic mapping studies. On the other hand, Rapid Reviews require less effort and resources and can deliver results in less time.

For the Rapid Review two search portals were queried. First, EBSCO Host was queried. Then, IEEExplore was queried. In both portals, the initial search string was <<Active AND teaching OR learning AND software engineering>> in all publications metadata. EBSCO Host was accessed through the project Portal Timbó made by ANII (Uruguayan National Agency for Innovation and Investigation).

The initial search string lead to many machine learning publications, where active learning has a different meaning. This required the search string to be rewritten. The new search string was <<Active AND teaching OR learning AND software engineering NOT machine>>, and the filters were applied to all publications metadata.

Timbó first returned 980000 publications. Only the first 60 publications sorted by relevancy were considered. IEEE returnes 13700 publications. Only the first 75 publications sorted by relevancy were considered.

Inclusion Criteria (IC) and Exclusion Criteria (EC) were applied to the preliminary results:

- IC1. The study is indexed or peer-reviewed
- IC2. The study focuses on Software Engineering higher education
- IC3. The study, if primary, is either an experience or a method presentation
- IC4. The study, if secondary, focuses on software engineering education
- EC1. The study was not published between 2018 and 2023
- EC2. The study was published on a language other than English, Spanish or Portuguese

The inclusion and exclusion criteria were applied. After this filtering stage, 16 studies found in Timbó were preserved, and 33 studies found in IEEE were preserved. The total amount of considered studies at this stage is 49 and are available in supplementary material for this study [15].

Data were extracted using a form. Out of each form the following data were extracted: title, link to search portal, main author affiliation, study type (primary experience or method, or secondary study), active learning strategies presented, year and authors names.

Active learning strategies found were analyzed following a data aggregation process. First, active learning strategies explicitly mentioned by the study were considered. For example <<Think-Pair-Square>>. Second, strategies using the same base methodology but called differently were analyzed individually. For example <<Games>>, <<Game-Based Learning>> and <<Gamification>>. In the examples mentioned, the base methodology is Games, but the way in which games are coordinated and integrated into the course differ.

In this study only one strategy name per base methodology is used. The name used is the most frequently name found in the studies. For example, <<Project-Based Learning>> and <<Projects>> were grouped under the name <<Project-Based Learning>>.

Finally, using the most commonly found name for base-methodologies, the number of occurrences of each strategy was counted across the studies.

4 Results

The first research objective (1) sought to identify and define what active learning strategies are used in Software Engineering education. 18 active learning strategies were found: Games, Project-Based Learning (PBL), Flipped, Peer-Based Learning, Problem-Based Learning (PrBL), Debates, Multimedia, Think-Pair, Service Learning, Experiential Learning, Concept Tests, Analogies, Virtual/Augmented Reality (VR/AR), Team-Based Learning (TBL), Just In Time Learning (JIT), Crowd-Based Learning, Competence-Based Learning and Blended Learning.

Table 1: Strategies occurrence frequency in Rapid Review

Strategy	Frequency
Games	16
Project-Based Learning (PBL)	14
Flipped	6
Peer-Based	6
Problem-Based Learning (PrBL)	5
Debate	4
Multimedia (MM)	3
Think-Pair (TP)	2
Service Learning	2
Experiential Learning (ELT)	2
Concept Test	2
Analogy	2
Virtual/Augmented Reality (VR/AR)	1
Team-Based Learning (TBL)	1
Just In Time Learning (JIT)	1
Crowd-Based Learning	1
Competence-Based Learning	1
Blended Learning	1

The second research objective (2) sought to know what application experiences exist for active learning strategies in Software Engineering education between 2018 and 2023. 49 studies were found. 41 works were primary studies reporting active learning strategies deployment, one of them still in progress, 6 studies presented methods for deploying active learning strategies, and 2 works were secondary studies. Table 1 presents how frequently were strategies found in the studies.

4.1 Strategies overview

A brief definition on each active learning strategy found is presented in the following section. Definitions were constructed based mainly on publications reporting the strategy. Some strategies are seldom defined in published academic literature, and consequently universities websites defining the strategies were visited. Many universities have centers specialized on education that share resources on pedagogical approaches and techniques.

4.1.1 Analogies

Learning by examples or analogies allows to hide specific details of certain contents. Therefore, it makes easier to abstract away from details shared by all observations. Analogies can help relate unknown domains to well-known domains and smoothly transfer knowledge from one situation to another [16]. Despite being an intuitive way of connecting ideas, analogies must be carefully tailored so as to avoid spreading misinterpretations.

4.1.2 Blended Learning

The principal characteristic of Blended Learning is the coordinated integration of in-person instructional methods with online ones, instructor-guided methods and student-guided ones, as well as synchronous and asynchronous methods [17]. This strategy combines advantages of diverse groups of learning activities. This way, Blended Learning allows for both instructors and students make decisions about what works best for them at given times and takes into consideration both parties needs [18].

4.1.3 Competence-Based Learning

Competence-Based Learning is a knowledge-based methodology which aims at clearly giving proof on what can individual achieve as a result of learning processes. Rather than focusing on how can specific learning results contribute to carrying out specific tasks, it focuses on how can multiple skills or learning results, as well as their interactions, be applied to individual tasks [19].

Competence-Based Learning is a methodology that focuses on authentic learning scenarios. Such scenarios require more than just technical knowledge. By adding this kind of situations to learning environments, non-technical or soft skills can thrive even more than they do in traditional settings.

4.1.4 Concept Tests

Concept Tests are short, informal and specific tests carried out during class time. Concept Tests are used by instructors to evluate to what extent do students comprehend foundational basis on any given topic. Concept Tests provide instructors with insights on how is the entire class performing, rather than individual students' performance [20]. They also provide opportunities to activate learning processes and receive immediate feedback.

4.1.5 Crowd-Based Learning

Crowd-Based Learning seeks leveraging knowledge amassed by crowds on the internet to boost learning [21]. Moreover, this framework aims at lifting common blockers in traditional learning settings. These settings have often drifted away from the industry and is restricted to academic sources of information.

Crowd-Based Learning can incorporate valuable tools like YouTube, GitHub, StackOverflow, DockerHub and HuggingFace among many others to introduce new knowledge in the classroom. This way, students not only can keep up the pace with the industry, but they can also discover Software Engineering from a new perspective.

4.1.6 Debates

Debates exist to allow groups of people to share solutions to problems by presenting their pros and cons. Debates are tightly related to discussions.

Debates in a learning setting are implemented to deeply engage students in meaningful discussions. Students are the main protagonists of debates in class. This way, debates are a student-centered active learning strategy [22] which promote divergent thinking and interpersonal skills.

4.1.7 Experiential Learning

Experiential Learning is a kind of learning in which students are directly in contact with the object being studied [23]. It is often compared to traditional learning: that learning involving reading, listening and writing, but never actually getting involved with the reality being studied.

The purpose of Experiential Learning is to facilitate sensory experience and contextualised actions [24]. Experiential Learning dialectics are based upon the idea of knowledge being created through experience transformation.

4.1.8 Flipped

Flipped Clasroom is a learning experience that involves students carrying out activities which are often carried out by instructors in traditional learning settings, and vice versa. Furthermore, it involves investing class time in activities often done out of class, and vice versa [25] [26].

Flipped Classroom make students tailors of synchronous class activities. It also permits students study at their own pace, fosters critical thinking and peer discussions. In Flipped Classroom settings learning is not constrained to what instructors decide to teach in traditional learning settings.

4.1.9 Games

Games are a well-established topic on education. Game-Based Learning is one of the most frequently heard concepts in learning settings. Game-Based Learning is any instructional design based on games. Games are activities characterized by players actively participating by interacting with game mechanics and win and lose conditions. Games often lead to learn and develop skills [27].

Games can be incorporated in instructional design even when not applying Game-Based Learning. Games can be included in any point in time during a course. Activities can also be gamified. Gamification involves turning an activity that is typically not a game into a game. Such an example is to transform a questionnaire into a game by adding a friendly competition where students are given a score based on how accurately they can answer to questions and how quickly they can do so.

4.1.10 Just In Time Learning

Just In Time Learning does not refer to any particular theory or learning method [28]. In comparison to traditional learning methods, Just In Time Learning has to do with informal, student-centered, self-paced and hands-on learning.

Just In Time Learning is not pre-designed or controlled in any way. Students define their own learning environment and adapt it according to their evolving needs. Another characteristic of this kind of learning is the reactivity of the learning process, which is permanently transformed according to self-defined student requirements.

4.1.11 Multimedia

Multimedia Content is content presented in multiple material supports. In the context of education, using multiple material supports boosts instructional methods by making them more effective. It also allows students to choose whatever support works best for them, giving them ownership of their learning process.

Modern supports are interactive [29], that is to say they are synchronous in relation to the learner's cognitive activity. One of the most relevant supports is video. Videos, along with interactive activities such as comments or questionnaires make a particular learning method called Active Video Watching [30].

Using multiple material supports for information do not necessarily entail active learning. Multimedia Content aid active learning as long as it is interwoven with activities that engage students with the learning process and content.

4.1.12 Peer-Based Learning

Peer-Based Learning is an educational practice in which students work along each other to achieve specific learning results [31]. Peer-Based Learning shares some characteristics with collaborative learning, which reunites diverse pedagogical frameworks involving shared effort.

4.1.13 Problem-Based Learning

Problem-Based Learning focuses on small groups of students working together to provide solutions to a problem. Instructors or tutors work together to aid groups as required, by helping the problem discovery, spotting key concepts, researching and sharing possible solutions to the problem, as well as reflecting on the topics covered [32].

In contrast to Project-Based Learning, Problem-Based Learning seeks to provide students with a process to learn new information, rather than applying previously acquired knowledge to build a new solution or product [33].

4.1.14 Project-Based Learning

Project-Based Learning seeks to enhance students understanding of a given topic by working on meaningful projects and developing products [34]. Projects involve the development of a product with well defined objectives and a well defined time-frame. Working on projects can help students deal with challenges offered by teamwork. It can also provide experience managing real-life scenarios complexity.

Project-Based Learning involves students actively participating and instructors being helpers. Project-Based Learning focuses on applying previously acquired knowledge towards a specific goal [33].

4.1.15 Service Learning

Service Learning is a pedagogical approach [35] in which students are lead to critically reflecting on how can their knowledge be brought to society in the shape of service [36]. Service Learning is characterized by deliberate and organized activities carried out by students in which previously assessed community needs are addressed. In such doing, students should reflect on their activities and understand what impact do they have in their communities.

Service Learning activities enhance curricular topics by providing a broader perspective on the object being study. It also enhances civic responsibility. All this should happen while also earning course credits [37].

4.1.16 Team-Based Learning

Team-Based Learning is an active learning strategy which seeks to apply knowldge through a series of activities involving both individual and teamwork as well as immediate feedback. Team-Based Learning starts with students preparation. Students study beforehand on their own. Then, the readiness assurance process is carried out. The readiness assurance process consists of questionnaires which are done first individually and then in teams. Teams remain the same along the entire course. Activities within the readiness assurance process provide immediate feedback. After the readiness assurance process, students work in teams applying the knowledge they acquired during the previous stages [38].

4.1.17 Think-Pair

Think-Pair is an activity which involves answering a question or finding a solution to a problem in stages. The first stage involves students individually looking for a solution to the problem. Second, students share their progress in pairs. Finally, pairs share their progress with the rest of the class [39].

Think-Pair is a base activity. There are many variations of it. One of the most relevant variations is Think-Pair-Square. In Think-Pair-Square, after doing the pairs activity, students also share their progress in groups of approximately 4 students.

4.1.18 Virtual Reality/Augmented Reality

Virtual Reality involved simulated representations with of a given phenomenon. These representations often seem real. Augmented Reality hosts such representations in a real-world setting, while Virtual Reality is completely fictitious.

The purpose of Virtual and Augmented Reality is to create an immersive experience, often lead by technology. In educational settings, both methods can be incorporated to instructional design in order to create sensory experiences, which can lead to almost authentic learning contexts [40].

5 Discussion

This sections presents a taxonomy shown on Figure 2 and 1 and may be used by instructors to choose strategies to be implemented according to learning results sought. According to this taxonomy, strategies may be considered general, Software Engineering specific, or both; or pedagogical frameworks, techniques, or both, according to how deep the strategy has consequences within a course.

There are existing taxonomies on teaching and learning strategies on engineering [41], but such taxonomies do not express the aim we seek in specific active learning terms.

5.1 Pedagogical frameworks and active learning techniques

The active learning strategies presented may impact the instructional design at different levels. Pedagogical frameworks are general groups of activities and philosophical conceptions applied at the highest level of abstraction within a course. Active learning techniques are specific activities applied within a well-defined time frame, although being repeatable. Some strategies can be considered both a pedagogical framework and a technique according to how it impacts the instructional design. Figure 1 exhibits what active learning strategies can be understood as frameworks or techniques.

Pedagogical frameworks may provide philosophical and conceptual support to a course design. They are considerably harder to implement. Techniques, on the other hand, can be interwoven with other active learning techniques and be introduced at a given point in time within a course.

An example of an active learning technique is debates. A debate may be implemented when a controversial content is presented to students so as to nourish them with their peers opinions and make them provide support to their ideas. Debates do not have to be implemented in topics where discussions may not be as enriching. Techniques are ideal for instructors willing to first introduce active learning strategies within their courses.

On the contrary, a pedagogical framework like TBL requires an entire course to be designed around it. Frameworks and techniques can coexist and complement each other, but the latter are more flexible and easier to deploy.

Many pedagogical frameworks have a technique associated to it. For example, game is the underlying technique behind gamification and GBL. The project is the underlying technique behind PBL. The same happens with problems and PrBL. Games, projects and problems can be added at any given point in time within a course, and it does not entail the course is designed according to GBL, PBL or PrBL.

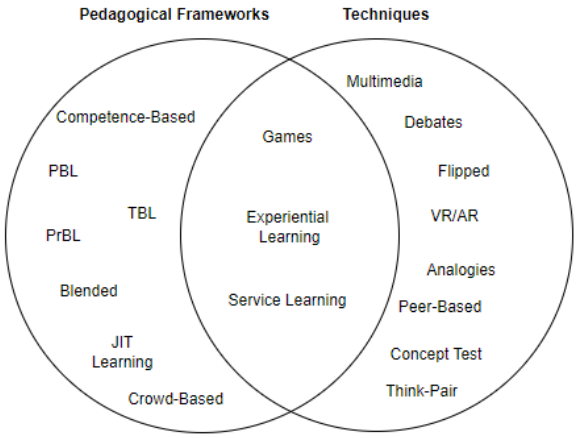

Figure 1: Active learning strategies classification as pedagogical frameworks or techniques.

The name given to each strategy found through the RR responded to what is the most frequently found name in the studies considered. Games, Experiential Learning and Service Learning are too broad as to be restricted to only one classification.

5.2 Individual and collaborative strategies

Some frameworks and techniques either promote or demand teamwork. There are no strategies that demand individual work. TBL is an example of a framework where teamwork is not optional. The same happens with PBL, PrBL, Debates, Crowd-Based, Peer-Based and Think-Pair. The rest of the strategies accept teamwork but do not demand it. It is assumed that problems in Software Engineering are complex enough as to demand teamwork.

Crowd-Based does not strictly require collaboration between peers, but leverages collective teamwork at a higher level of abstraction, therefore teamwork is considered not to be optional. It demands interactions between individuals geographically and chronologically disperse.

5.3 Strategies consisting of software activities and general activities

Some strategies involve software engineering specific activities, such as requirements elicitation, development, testing, validation, deployment and documentation. Some strategies involve general learning activities, such as reading, self-evaluation, communication and so on. This classification is shown on Figure 2 and may be used by instructors to choose strategies to be implemented according to learning results seeked.

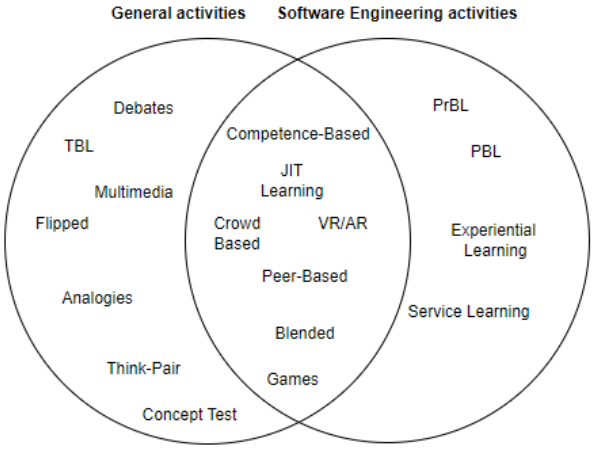

Figure 2: Active learning strategies classification as involving general or Software Engineering activities.

Higher education often balances theoretical knowledge and hands-on experience. This classification may be used by instructors to choose active learning strategies according to the desired educational outcome.

Strategies like Debates, Analogies and Concept Tests do not require necessarily require carrying out Software Engineering activities. On the other hand, PrBL, PBL, Experiential Learning and Service Learning do entail some kind of Software Engineering activity when deployed within a Software Engineering learning setting.

Some strategies like Peer-Based Learning, Games and Competence-Based Learning do not necessarily require the execution of Software Engineering specific activities, but such activities may be blended in its implementation.

5.4 Combining strategies

Some works gathered via the RR combine different active learning strategies. Pedagogical frameworks are often combined with active learning techniques. Such is the case for PBL (out of 14 times applied 8 times was accompanied by other strategies), Peer-Based Learning (out of 6 times applied 4 times was accompanied by other strategies) and PrBL (out of 5 times applied, 3 times was accompanied by other strategies).

Experiential Learning, TBL, JIT and Blended were accompanied by other strategies all the times they were reported. The rest of the pedagogical frameworks (Service Learning and Competence-Based Learning) were not reported to be combined with other strategies.

Some techniques are intrinsically similar to each other, like Peer-Based Learning and Think-Pair. Some others complement each other, such as PrBL and Experiential Learning, which have been reported to be applied together.

Some other combinations are less evident, like Games and PBL. Such combinations are the result of a specific instructional design.

6 Deployment and Evaluation of Active Learning strategies

In this section, we present a pedagogical case study focusing on the deployment and evaluation of active learning strategies. We analyze the systematic and ongoing deployment of different pedagogical strategies in all mandatory courses of the Software Engineering Department at Universidad ORT Uruguay. In 2023 the authors sought to systematically collect quantitative and qualitative data of both active and traditional learning strategies deployment. We use two main sources: documents with courses description (learning goals, syllabus, evaluation) and a group of semistructured interviews with teachers to collect insights and lessons learned about the deployment and evolution of the TBL methodology in three courses.

6.1 SE courses pedagogical framework

The Software Engineering degree takes 10 semesters. Each course lasts 16 weeks. In each course, students can earn up to 100 points. Students who obtain 70 points or more approve. Those who do not achieve 70 points are split into two groups: those who achieve a certain minimum threshold of points on specific activities are given a second chance to take part in one of the courses' evaluations, and according to the new result, they approve or not. Those who do not meet the minimum threshold of points on individual activities defined at the beginning of the course must take the whole course over again.

The Software Engineering Department has 13 courses, which are described as follows.

- The course Software Engineering Foundations (FIS) is planned to be taken in semester 4 and its main objective is to obtain foundational Software Engineering skills.
- The course Application Design 1 (DA1) is planned to be taken in semester 5 and its main objective is to obtain foundational Object-Oriented Design skills.
- The course Application Design 2 (DA2) is planned to be taken in semester 6 and its main objective is to obtain advanced Object-Oriented Design skills on full-stack applications.
- The course Agile Software Engineering 1 (ISA1) is planned to be taken in semester 6 and its main objective is to obtain foundational Agile Software Engineering skills using Scrum.
- The course Agile Software Engineering 2 (ISA2) is planned to be taken in semester 7 and its main objective is to obtain advanced Agile Software Engineering skills in DevOps organizations.
- The course Software Architecture (AR) is planned to be taken in semester 7 and its main objective is to obtain skills on Software Architecture.
- The course Software Architecture in Practice (ASP) is planned to be taken in semester 8 and its main objective is to practice skills on Cloud Software Architecture.

- The course Technology-Based Product Design (DPBT) is planned to be taken in semester 8 and its main objective is to obtain foundational skills on product design and management.
- The course Human-Computer Interaction (IHC) is planned to be taken in semester 8 and its main objective is to obtain foundational development skills on mobile applications.
- The course User Centered Design (DCU) is planned to be taken in semester 8 and its main objective is to obtain foundational skills on usability and user-centered design approaches.
- The course Communication and Conflict Management (GCCM) is planned to be taken in semester 9 and its main objective is to obtain foundational soft skills on conflict management.
- The course Software Development Team Skills (HEDS) is planned to be taken in semester 9 and its main objective is to obtain foundational soft skills on software development teams.
- The course Management Skills for Software Projects (TNEP) is planned to be taken in semester 9 and its main objective is to obtain foundational soft skills on management of software projects.

Universidad ORT Uruguay's Engineering Faculty institutionally offers three types of evaluations: exams, projects and activities.

Exams are traditional evaluations, most often in-person, synchronous written evaluations. Exams have little of active learning, and seek to measure individual performance on students.

Projects are practical assignments done by groups of 2 or 3 students, with a fixed start and end date, usually split apart by a couple weeks. In most cases, projects seek to allow students to put into practice software development skills. Projects are considered deployment of active learning strategies at Universidad ORT Uruguay.

Some projects have been scoped out enough so that students can go straight and implement solutions. In such cases these projects are aligned with Project-Based Learning.

Some projects are fairly open-ended and thus require students to scope out the problem. In such cases, these projects are in line with Problem-Based Learning, as well as Project-Based Learning.

For example, Application Design 2 has two projects. The first one entails implementing a backend service on ASP.NET Core using C#. There are approximately 10 CRUD requirements to be implemented using Test-Driven Development and verified using Postman. Students have about 5 weeks to achieve this in groups of 3. The second project entails adding some new CRUD requirements as well as a frontend application using Angular. Students have about 4 weeks to achieve this goal. Each project delivery has mandatory software quality goals, such as maintainability, design documentation, and functional testing.

Activities are a heterogeneous group of techniques. Each course takes advantage of activities' flexibility and adapts it to fit the learning results intended by the course. Some activities are active, some are traditional, some are hybrid.

Within activities many active learning strategies can be found: Debates, Team-Based Learning, Flipped, Concept Tests, Games and Project-Based Learning. There are also activities with traditional strategies like plain questionnaires.

One relevant example of activities in Agile Software Engineering 2 activities. In this course, activities give up to 40 points. 20 of such points are obtained via TBL's Readiness Assurance Tests (RATs) which are considered active learning activities due to their flipped nature and because they happen within an active learning pedagogical framework. The other 20 points can be obtained via TBL's application exercises, which are also considered active learning techniques.

6.2 Analysis of active learning practices

The number and type of active learning activities deployed in each of the courses has some degree of variation. Moreover, in each academic term, lecturers could make changes to course activities and evaluations, although these changes are usually gradual and minor (in terms of the total points of the course). To analyze globally the impact of different active learning approaches, we focused on the relative weight of these type of strategies in each course. We excluded optional learning activities, considering only mandatory graded activities that are neccesary to approve the course.

The evaluations carried out in the Department have been classified according to 2 criteria: first, using the individual versus collaborative classification presented previously in this study. Second, classifying the evaluation activities are active or traditional. Activities are considered active if there is an active learning strategy as presented in this paper backing it up. Out of 100 points for each course, each individual point has been clasified according to whether if it can be obtained by students via active or traditional evaluation activities. Table II presents the results. Data about the courses was collected via interviewing lecturers and revising official documents.

Table 2: Comparison of Learning Strategies by Category

Acronym	Individual (%)	Group (%)	Active (%)	Traditional (%)
FIS	40	60	60	40
DA1	50	50	50	50
DA2	50	50	50	50
ISA1	40	60	60	40
ISA2	48	52	80	20
AR	46	54	75	25
ASP	40	60	80	20
DPBT	40	60	60	40
IHC	30	70	70	30
DCU	50	50	50	50
TNEP	40	60	60	40
GCCM	40	60	60	40
HEDS	40	60	60	40
Department	**43**	**57**	**63**	**37**

6.3 TBL evolution and lessons learned

Another relevant activity conducted in order to improve some Software Engineering courses involved studying the evolution of TBL deployment across three different courses: ISA1, ISA2 and AR. In order to achieve this, semi-structured interviews were done to 6 lecturers involved in TBL.

Two main results were drawn from this analysis. First, events timelines were constructed to trace back the evolution of TBL for each subject. Agile Software Engineering courses deepened their deployment of TBL iterating on the score students can obtain from its activities. Software Architecture, on the contrary, no longer applied TBL per se. In spite of this, AR still gets inspiration from TBL by doing flipped activities close to RATs.

The second result is an analysis carried out by processing the interviews and grouping up ideas on the following areas: lecturers' experience, lecturers' perspectives on student experience, challenges and lessons learned using TBL.

On lecturers' experience the main takeaways are that TBL drastically shifts the attention of the lecturer from actually lecturing towards becoming an enabler. Lecturers do also have a greater time in class than in those classes that do not use TBL. Another key takeaway is that TBL activities like preparing RATs and application exercises are really high-maintenance.

On lecturers' perspective on student experience the main takeaways are that it is not easy for students to first face TBL. In spite of this, they get the hang of it fairly quickly. Lecturers do also report that TBL broadens the gap between students to actively go to class and deliberately take part in it to those who do not attend to class, since it is not compulsory at Universidad ORT Uruguay.

On challenges, the three main takeaways are that TBL activities are really high maintenance. It is also challenging to calibrate scoring. Finally, it is hard to merge the tech stack used by a given course with TBL activities.

On lessons learned, TBL and active learning strategies seem to have deepend student's knowledge on course content than before. The second lesson learned is that activities that are not scored are less attractive to students and therefore less effort is invested on them. Finally, lecturers report that they feel in-person activities are considerably more productive than remote or even hybrid ones.

7 Threats to validity

Groups of similar strategies like Games, Gamification and Game-Based Learning were reduced to a single strategy for the sake of brevity, choosing the one more often found within the considered studies. All strategies in all of their formats are relevant within the context of Active Learning in Software Engineering. This reduction conditioned all the classifications presented.

All the strategies classification presented in this work were carried out by a single author, but peer-reviewed by other researchers.

The Rapid Review method trades off between velocity and depth. This Rapid Review provided results leveraged by decision takers to adjust SE courses in a higher education setting. Reach and precision were less strict than a traditional Systematic Mapping Study.

8 Conclusions

Software engineering education can take advantage of active learning in higher education. Games and projects are the most frequently found strategies in literature. These resources and many others can sporadically be introduced in instructional design. They can also determine the pedagogical framework within the instructional design.

Active learning strategies promote both knowledge and competencies development. These elements are all keystone to the growth required by software engineering students in higher education. Leveraging both traditional knowledge and competencies can definitely contribute to shrinking the gap between the software engineering students skill-set and the industrial sector expectations on those students.

Active learning strategies used are diverse currently. Being that said, the majority of the strategies presented in this work share certain characteristics. For instance, all active learning strategies involve teamwork to a certain extent. Some strategies demand teamwork while others just promote it.

There are strategies that involve software development activities and strategies that involve other intellectual activities such as debating or writing. Active learning strategies that involve software engineering and development activities are the ones that directly contribute to shrinking the gap between the industry expectations on students experience working on Software Engineering challenges and the actual experience amassed by students during their education.

Active learning can be deployed within any higher education learning environment. There are straightforward approaches that are referred as techniques in this work. Techniques are point-in-time specific and repeatable. There are other strategies with a deeper meaning and higher deployment cost referred as pedagogical frameworks in this work. Pedagogical frameworks span an entire course and dictate what other activities are compatible with it. In any case, active learning strategies can and should be combined to customize learning environments in software engineering.

References

[1] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). Manifesto for Agile Software Development. In Manifesto for Agile Software Development. http://www.agilemanifesto.org/

[2] Borges, G. & Souza, R. Skills development for software engineers: Systematic literature review. *Information And Software Technology*. **168** pp. 107395 (2024,4)

[3] G. Matturro and M. Solari, Soft skills in software engineering in higher education. An initial study of its state in Latin America, in XXII Conferencia Iberoamericana de Software Engineering (CIbSE 2019), La Habana, Cuba, 2019.

[4] Garousi, V., Giray, G., Tuzun, E., Catal, C. & Felderer, M. Closing the Gap Between Software Engineering Education and Industrial Needs. *IEEE Software*. **37**, 68-77 (2020,3)

[5] A. W. Chickering y Z. F. Gamson, Appendix A: Seven principles for good practice in undergraduate education, New Directions for Teaching and Learning, vol. 1991, no 47, pp. 63–69, sep. 1991, doi: 10.1002/tl.37219914708.

[6] C. C. Bonwell y J. A. Eison, Active Learning: Creating Excitement in the Classroom, 1991. [En línea]. Disponible en: www.eric.ed.gov

[7] M. Prince, Does Active Learning Work? A Review of the Research, Journal of Engineering Education, vol. 93, no 3, pp. 223–231, jul. 2004, doi: 10.1002/j.2168-9830.2004.tb00809.x.

[8] L. Springer, M. E. Stanne, y S. S. Donovan, Effects of Small-Group Learning on Undergraduates in Science, Mathematics, Engineering, and Technology: A Meta-Analysis, Rev Educ Res, vol. 69, no 1, pp. 21–51, mar. 1999, doi: 10.3102/00346543069001021.

[9] J. I. Irabedra, M. Solari and G. Mousqués, Active Learning in Software Engineering: A Rapid Review, 2024 L Latin American Computer Conference (CLEI), Buenos Aires, Argentina, 2024, pp. 1-8, doi: 10.1109/CLEI64178.2024.10700583.

[10] Marques, M., Quispe, A. & Ochoa, S. A systematic mapping study on practical approaches to teaching software engineering. *2014 IEEE Frontiers In Education Conference (FIE) Proceedings*. pp. 1-8 (2014,10)

[11] Daun, M., Grubb, A., Stenkova, V. & Tenbergen, B. A systematic literature review of requirements engineering education. *Requirements Engineering.* **28**, 145-175 (2023,6)

[12] Kondo, H. & Hazeyama, A. Systematic Literature Review on Educational Effectiveness of Project-Based Learning for Software Development. *2022 29th Asia-Pacific Software Engineering Conference (APSEC).* pp. 584-585 (2022,12)

[13] L. Cernuzzi, "Autumn of Code UC: an Experience Teaching Software Engineering Contributing to Open Source Projects," 2024 L Latin American Computer Conference (CLEI), Buenos Aires, Argentina, 2024, pp. 1-9, doi: 10.1109/CLEI64178.2024.10700211.

[14] B. Cartaxo et al., Software Engineering Research Community Viewpoints on Rapid Reviews," in 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), IEEE, Sep. 2019, pp. 1–12. doi: 10.1109/ESEM.2019.8870144.

[15] J. I. Irabedra, M. Solari, G. Mousqués. "Supplementary Material - Active Learning in Software Engineering A Rapid Review". Zenodo, jul. 19, 2024. doi: 10.5281/zenodo.12775497.

[16] V. Lian, E. Varoy, and N. Giacaman, Learning Object-Oriented Programming Concepts Through Visual Analogies," IEEE Transactions on Learning Technologies, vol. 15, no. 1, pp. 78–92, Feb. 2022, doi: 10.1109/TLT.2022.3154805.

[17] A. Alammary, Blended learning models for introductory programming courses: A systematic review," PLoS One, vol. 14, no. 9, p. e0221765, Sep. 2019, doi: 10.1371/journal.pone.0221765.

[18] A. Manasrah, M. Masoud, Y. Jaradat, M. Irshaidat, N. A. Shaban, and A. Zerek, Students Engagement in Blended Learning: Evidence from Moodle A Case Study from Engineering Courses," in Proceeding - 2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering, MI-STA 2023, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 435–439. doi: 10.1109/MI-STA57575.2023.10169635.

[19] M. Jaanus, K. Umbleja, A. Udal, and K. Parnamets, Integrated labs for electrical engineering courses in competence based learning environment-practical experience," 2019 Electric Power Quality and Supply Reliability Conference and 2019 Symposium on Electrical Engineering and Mechatronics, PQ and SEEM 2019, Jun. 2019, doi: 10.1109/PQ.2019.8818266.

[20] Carnegie Mellon University, Using Concept Tests," `https://www.cmu.edu/teaching/assessment/assesslearning/concepTests.html`. Accessed: Feb. 03, 2024. [Online]. Available: `https://www.cmu.edu/teaching/assessment/assesslearning/concepTests.html`

[21] X. Mao, Y. Lu, L. Yin, T. Wang, and G. Yin, Model and Practice of Crowd-Based Education," in Web and Big Data, 2018, pp. 293–305. doi: 10.1007/978-3-030-01298-4_25.

[22] Y. Choi and E. Kim, A case study on the evaluation of discussion and debate learning effectiveness in a dental hygiene ethics class," European Journal of Dental Education, vol. 26, no. 2, pp. 223–231, May 2022, doi: 10.1111/eje.12690.

[23] D. A. Kolb, Experiential Learning Experience as the Source of Learning and Development. Prentice-Hall, 1984.

[24] A. Shore and T. Dinning, Developing student's skills and work readiness: an experiential learning framework," Journal of Work-Applied Management, vol. 15, no. 2, pp. 188–199, Sep. 2023, doi: 10.1108/JWAM-02-2023-0016.

[25] C. Paez-Quinde, A. Chasipanta-Nieves, C. A. Hernandez-Davila, and J. Arevalo-Peralta, Flipped classroom in the meaningful learning of the students of the Basic Education Career: Case study Technical University of Ambato," in 2022 IEEE Global Engineering Education Conference (EDUCON), IEEE, Mar. 2022, pp. 785–789. doi: 10.1109/EDUCON52537.2022.9766792.

[26] Y. Dong, H. Yin, S. Du, and A. Wang, The effects of flipped classroom characterized by situational and collaborative learning in a community nursing course: A quasi-experimental design," Nurse Educ Today, vol. 105, p. 105037, Oct. 2021, doi: 10.1016/j.nedt.2021.105037.

[27] M. Videnovik, T. Vold, L. Kiønig, A. Madevska Bogdanova, and V. Trajkovik, Game-based learning in computer science education: a scoping literature review," Int J STEM Educ, vol. 10, no. 1, p. 54, Sep. 2023, doi: 10.1186/s40594-023-00447-2.

[28] D. C. Brandenburg and A. D. Ellinger, The Future: Just-in-Time Learning Expectations and Potential Implications for Human Resource Development," Adv Dev Hum Resour, vol. 5, no. 3, pp. 308–320, Aug. 2003, doi: 10.1177/1523422303254629.

[29] Krismadinata, U. I. Kurnia, R. Mulya, and U. Verawardina, The Interactive Multimedia Learning for Power Electronics Course," International journal of online and biomedical engineering, vol. 18, no. 7, pp. 44–56, 2022, doi: 10.3991/ijoe.v18i07.30029.

[30] M. Galster, A. Mitrovic, and M. Gordon, Toward Enhancing the Training of Software Engineering Students and Professionals Using Active Video Watching," 2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET), 2018.

[31] A. M. O'Donnell and A. King, Cognitive perspectives on peer learning. in The Rutgers Invitational Symposium On Education Series. Mahwah, NJ, US: Lawrence Erlbaum Associates Publishers, 1999.

[32] S. A. Nicolaou, A. Heraclides, C. S. Constantinou, S. Loizou, and D. J. Gillott, One size doesn't fit all: PBL tutor training and development," Interdisciplinary Journal of Problem-based Learning, vol. 15, no. 2, Dec. 2021, doi: 10.14434/IJPBL.V15I2.30267.

[33] S. Williams van Rooij, Scaffolding project-based learning with the project management body of knowledge (PMBOK®)," Comput Educ, vol. 52, no. 1, pp. 210–219, Jan. 2009, doi: 10.1016/j.compedu.2008.07.012.

[34] J. A. Hurtado, A. C. Useche, and B. S. Masiero, Project-Based Learning: Authentic Engineering Assessment Supported by Model Design," International Journal of Engineering Pedagogy (iJEP), vol. 13, no. 6, pp. 17–32, Sep. 2023, doi: 10.3991/ijep.v13i6.38539.

[35] V. Duarte, M. Cleveland-Slimming, C. Vidal, and S. Contreras, Information System Development by Using Agile Teamwork and Service-Learning," IEEE Transactions on Education, vol. 66, no. 5, pp. 431–441, Oct. 2023, doi: 10.1109/TE.2023.3310243.

[36] L. M. Shea, D. Harkins, S. Ray, and L. I. Grenier, How Critical is Service-Learning Implementation?," Journal of Experiential Education, vol. 46, no. 2, pp. 197–214, Jun. 2023, doi: 10.1177/10538259221122738.

[37] S. R. Bandi, G. Joshi, A. Shettar, and R. Kandakatla, A Systematic Literature Review on Faculty Learning in Service-Learning," in 2023 IEEE Global Engineering Education Conference (EDUCON), IEEE, May 2023, pp. 1–10. doi: 10.1109/EDUCON54358.2023.10125223.

[38] R. E. Levine and P. D. Hudes, How-to Guide for Team-Based Learning. Cham: Springer International Publishing, 2021. doi: 10.1007/978-3-030-62923-6.

[39] Brown University, Interactive Classroom Activities," https://www.brown.edu/sheridan/teaching-learning-resources/teaching-resources/classroom-practices/active-learning/interactive. Accessed: Feb. 03, 2024. [Online]. Available: https://www.brown.edu/sheridan/teaching-learning-resources/teaching-resources/classroom-practices/active-learning/interactive

[40] L. Kong, Empirical Study on the Effects of the Application of Virtual Reality to Experiential Education on Students' Learning Attitude and Learning Effectiveness," Revista de Cercetare si Interventie Sociala, vol. 73, pp. 288–298, Jun. 2021, doi: 10.33788/rcis.73.18.

[41] B. Zhong, X. Liu, L. Xia, and W. Sun, "A Proposed Taxonomy of Teaching Models in STEM Education: Robotics as an Example," *SAGE Open*, vol. 12, pp. 215824402210995, May 2022, doi: 10.1177/21582440221099525.