Learning Together in Global Classrooms: Student Engagement Through Collaborative Activities and Games

Simona Vasilache
Institute of Systems and Information Engineering
University of Tsukuba
Tsukuba, Japan
e-mail: simona@cs.tsukuba.ac.jp

Abstract - **Software engineering education increasingly takes place in classrooms where students from diverse cultural backgrounds learn side by side. While this diversity enriches the learning environment, it also introduces challenges in communication, collaboration, and engagement. This paper reports on experiences from teaching a software engineering course with a highly international student cohort, focusing on the role of classroom activities as cross-cultural bridges. Active learning strategies, such as collaborative problem-solving exercises and in-class games, were employed to encourage participation, foster mutual understanding, and develop teamwork skills essential to professional practice. Drawing on classroom observations and student feedback, this work highlights how such activities can mitigate cultural barriers and create opportunities for students to appreciate different perspectives. Findings suggest that collaborative exercises not only improved engagement but also facilitated learning and strengthened students' sense of belonging in a multicultural environment. The paper argues that designing engaging classroom activities, where students from different cultures work together towards achieving a common goal, is crucial for preparing students to work in global software engineering contexts, where cross-cultural collaboration is essential.**

Keywords - software engineering; student engagement; active and collaborative learning; multicultural environments.

I. INTRODUCTION

We live in a world which relies heavily on software systems. Although a relatively young discipline (57 years since it was framed as an engineering discipline in its own right, i.e., in 1968), software engineering provides systematic methods for designing, developing and maintaining software systems that underpin our modern society [1]. In higher education, software engineering related subjects are well established and offered by an increasing number of institutions. According to CS2023 [2], a computer science curricular guidelines document (published by the Joint Task Force of Computer Science Curricula, which comprises ACM, IEEE-CS, and the Association for the Advancement of Artificial Intelligence), "since 2013, the focus of curricular design has moved from what is taught (a knowledge model) to what is learned (a competency model)" [2]. Thus, educators teaching software engineering subjects must equip students not only with technical concepts, the so-called "hard skills", but also with a range of interpersonal and professional "soft skills".

In 2024, a systematic literature review of skills development revealed 33 essential soft skills that educators must teach future software engineers [3]. The top 5 soft skills revealed were communication, teamwork, organization, leadership and learning. They were followed by creativity, critical thinking, analytical skills, problem solving and professionalism. To help with acquiring important abilities like teamwork, communication, and analytical skills, collaborative learning is a very useful tool in a software engineering classroom. This strategy makes sure that students "collaborate" in order to achieve a task and this fosters development of various soft skills. Through collaborative learning, educators can simulate real-world professional settings, where software engineers must work together to develop software applications.

In their learning experience, students' performance is shaped by a wide range of factors, with motivation standing out as one of the most important. In the context of engineering disciplines, motivation has been described as "particularly critical" [4], given the complexity of the skills and knowledge students are expected to acquire. In response, educators have increasingly turned their attention to innovative pedagogical approaches that can increase and sustain motivation. Among these, game-based learning has gained significant momentum in recent years due to its demonstrated potential to enhance engagement, encourage active participation, and ultimately improve learning outcomes [5].

While studying, learners are often part of international classrooms. As a matter of fact, multicultural environments are now a common feature of workplaces, academia, and everyday life. Teaching in such settings presents not only specific challenges [6] but also distinct advantages [7]. A multicultural classroom can serve as a microcosm of distributed development teams, achieving the goal of providing global software engineering education. While students learn together in the same physical space, they simultaneously gain insight into how they may need to collaborate in the future with colleagues from different cultural backgrounds, each bringing diverse values, behaviors, and working styles [7].

In our previous works ([1], [8]), we highlighted some of the challenges of teaching an introductory software engineering course to a multicultural group of graduate

students in a Japanese university. This paper extends our work with further exploration of how student engagement was achieved through collaborative activities and game-based learning, along with lessons learned during this course.

The remainder of this paper is organized as follows. Section II describes the background of our work, as well as related work. A description of our course is provided in Section III, followed, in Section IV, by an illustration of collaborative learning and game-based learning as they were employed in our course. Section V includes a discussion and lessons learned during our study. Finally, section VI provides conclusions and directions for future work.

II. Background and related work

This section examines considerations on collaborative learning, game-based learning and global software engineering, along with related work outlining recent development in these fields.

A. Collaborative Learning

Collaborative learning is an important and useful tool promoting engagement of learners. Despite a generally accepted lack of consensus on the rigorous definition of the term [9], it has certain widely recognized characteristics. As an educational approach to teaching and learning, it involves learners "collaborating" (i.e., working together) to achieve a common goal – solve a problem, complete a task or create a product [10], at the same time progressing individually during the learning process. Participants are challenged socially and emotionally while listening to different perspectives and they may often be required to defend their ideas [9]. Fundamentally, similarly to active learning, collaborative learning promotes active student participation [11].

In international settings such as multicultural classrooms, collaborative learning is particularly valuable for fostering awareness of cultural differences, exposing students to diverse perspectives, and strengthening mutual understanding, as they work together toward solutions acceptable to the entire group. It can also support the development of communication and social skills, by offering a safe and structured environment to interact with others [11].

Furthermore, collaborative learning helps prepare students for their future workplaces: it improves communication, negotiation and teamwork skills – all very important in software engineering, where projects are often team-based. This makes collaborative learning an essential tools for educators in software engineering courses, particularly those that include intensive team projects.

B. Game-Based Learning

In game-based learning (GBL), the focus is on acquiring knowledge and skills through gameplay. Whether digital or non-digital, specific games are designed to achieve certain educational goals. This strategy has drawn increased attention, in various disciplines, including software engineering. In their work, Garcia et al. [12] conducted a systematic literature review of the effects of GBL in acquisition of soft skills in undergraduate software engineering courses. Their review shows that researchers have recognized the effectiveness of using GBL in teaching and learning various software engineering topics. This is largely due to the games' intrinsic features, which make them attractive to students and improve their motivation [12]. Moreover, numerous researchers recognize that games contribute to the development of "soft skills", like teamwork and communication ([13], [14], [15]). Furthermore, Marti-Parreno et al. [16] argue that GBL can enhance the delivery of learning content by allowing students greater control over their own learning process during gameplay. By adopting a GBL approach, students are able to apply classroom concepts in practice, thereby reinforcing and deepening their understanding of those concepts [16].

GBL is used not only in academic environments, with students, but also in industry, for training employees. To give an example, the work of O'Farrell et al. [17] shows how it was used for training employees on the Scale Agile Framework, [18], through a 3D game named PlaySAFe. This study showcased various benefits of GBL, like "allowing newcomers a quick and efficient means to learn and understand the practical groundwork of SAFe in advance of learning more theoretical concepts in conventional training" [17].

Overall, GBL offers an effective pedagogical approach in software engineering by enhancing motivation and enabling students to apply theoretical concepts in practical, interactive contexts.

C. Global Software Engineering

Globally distributed software projects are widespread in today's world and special skills in communication across different locations and time zones need to be learned early by software engineering students. An increasing number of universities are incorporating global software engineering into their curricula, yet its adoption remains limited. In their work, Beecham et al. [19] emphasize the need for global software engineering education and summarize several global software engineering education related challenges and proposed solutions. According to Schmiedmayer et al., [20] there are two main options to make students aware of global software engineering challenges and equip them with the necessary skills to deal with them: teachers can simulate a global software project in a classroom setting, or they can arrange a genuine global software engineering project; the latter option comes with all the organizational challenges of a distributed organization and infrastructure [20].

In terms of learning and applying soft skills, a multicultural classroom offers a suitable environment to simulate some of the challenges of a globally distributed project, as exposed by some educators' work. For instance, the work in [21] shows how the online environment brought by the Covid-19 pandemic provided an opportunity to test mini-models of distributed teams in software engineering, in the context of a multicultural classroom. In such a setting, while working together on achieving various tasks in the classroom, students learn how to collaborate with colleagues from different countries, each with their own culture, language and specific expectations.

III. COURSE DESCRIPTION

This section describes the setting, composition and content of the course that constitutes the subject of this paper.

A. Course Setting and Class Composition

This paper is based on the experience of teaching an introductory software engineering course named "Principles of Software Engineering", as it was offered in its latest edition, in the spring semester of 2024. This course has been taught annually as an elective course in the Master's Program in Computer Science at the University of Tsukuba in Japan, between 2016 and 2022. Since 2023, it has been held once every two years, alternating with a computer ethics course; it takes place in even number years, thus no class was held in 2023. The length of the course is 10 weeks, with 3 hours held every week; if completed successfully, students obtain two academic credits for it. The grading is based on the submission of a final report, which the students have approximately 3 weeks to complete.

The course aims to familiarize students with the fundamental principles of software engineering and to highlight its importance as a modern engineering discipline. Core topics include software development models and life cycles, agile methodologies, requirements engineering, user interface design, verification and validation techniques, project planning and management, software engineering tools such as IDEs and UML, as well as the business aspects of software development. The 2024 edition of the course saw 105 participants: 35 students in their first year and 70 students in their second year of master's course. They were a mixture of Japanese students (43) and international students (49 regular students and 13 exchange students, coming from a total of 15 different countries). Most students belong to the computer science department; only 4 students belong to different departments.

It is worth mentioning that this course started with 15 participants in 2016 and reached 105 enrolled participants in 2024 (it generally grew every year, apart from a steep decline in 2020, during the beginning of Covid-19 pandemic). Table I shows the total number of students, along with the number and percentage of international students enrolled in each of the 8 editions of the course. As can be observed, international students usually represent between 50% and 70% of the total number of students. Notably, this course is being held in English (and the instructor is non-Japanese, as well). This feature is responsible for attracting comparatively many international students, who have fewer course choices of courses held in English and for whom, often, taking classes held in Japanese is challenging. All communications, teaching, class materials, plenary discussions are held using English. However, students are allowed to submit their assignments (or any other feedback on the learning-management system) in Japanese. Moreover, group discussions are allowed in any language, as long as it is understood by all the participating members. Importantly, using a language other than English is only allowed in class verbally, but never in writing (since shared documents are often seen by all students, who need to understand them). Besides English, Japanese was the most spoken language in class; there were several instances in which certain small groups spoke Chinese or French while performing their class tasks. Last, but not least, reports or assignments, for which the access is restricted to the submitting student and the instructor, could be submitted either in English or in Japanese.

TABLE I. INTERNATIONAL STUDENT ENROLLMENT IN SOFTWARE ENGINEERING COURSE

	Total number of students	*Number of international students*	*Percentage of international students*
2016	15	9	60%
2017	26	18	69.2%
2018	35	24	68.5%
2019	66	33	50%
2020	34	28	82.3%
2021	53	37	69.8%
2022	66	33	50%
2024	105	62	59%

B. Course Content and Class Flow

The course combined a variety of teaching methods, including discussions, brainstorming activities, games, micro-projects, and instructor-led lectures. More specifically, classes usually started with a 5-minute warm-up activity (which could be a short discussion on a recent science or technology piece of news). This was followed by a repeated combination of lecture (in which the instructor explained a new topic), discussions among students (either in groups or with the whole class), and various activities and/or games. Figure 1 illustrates how lecture, class activities and games are interconnected during each class. The lecture part is always followed by discussions, which are closely connected to class activities and often games, as well. The cool-down part is mostly made-up of summarizing/concluding discussions. Notably, discussions were part of the class since its inception (in 2016); gradually, activities and games were introduced in subsequent years, with the variety and number of items increasing every year.

Throughout the course, the instructor used every opportunity to gather feedback from the students, either through informal discussions (during the break time or after class) or by means of written feedback, submitted trough the learning management system (LMS) used in the course, i.e., *manaba* [22]. Following some of the longer activities, students were invited to provide extensive feedback on aspects such as strengths and weaknesses, language or group preferences, and general impressions.

During the last class, the link to a comprehensive survey was distributed, to which 30 students responded, out of a total enrollment of 105. The instructor attributes this relatively low response rate to timing, as the survey was administered at the end of the course, when many students felt they had already provided sufficient feedback during earlier sessions. Finally, only 31 students responded to the end-of-course evaluation questionnaire provided by our university.

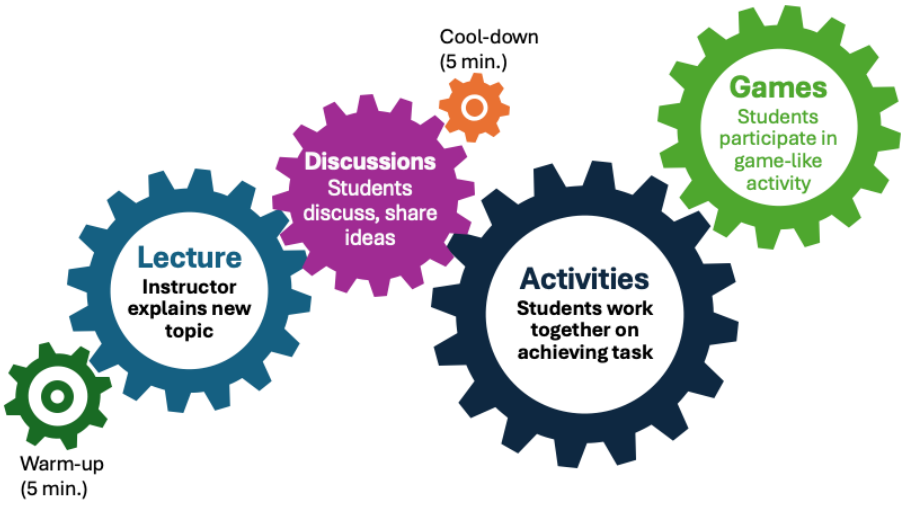

Figure 1. Visual representation of inter-connected class components of introductory software engineering course

IV. Implementation of Collaborative Learning and Game-Based Learning

This section illustrates two class activities: the first one shows the implementation of collaborative learning, whereas the second one provides an example of deploying game-based learning.

A. Class Project: Requirements Elicitation and Requirements Specification

The ability to work and collaborate/communicate successfully in a team is a crucial skill for a future software engineer. One of the most effective ways to develop teamwork and communication skills is to include a project in the coursework where students must work in teams. This work in groups is "essential in active approaches that are based on real-world problem-solving practices" [23].

To achieve this purpose, one of the classes during this course was dedicated to a project in which teams of students were responsible with creating a (simple) requirements document for a given application. This activity covered requirements elicitation and requirements specifications for a given software application.

The "Project" function of the LMS was used to facilitate work in teams, as well as sharing the teams' work with the whole class. The activity was divided into 3 main parts, with 6 tasks in total. In the first part, requirements elicitation took place; in the second part, the requirements document was created; in the last part, feedback on this document was collected among team members. Finally, the created requirements documents and everything included in the "Project" was shared with the whole class (all the students had access to all the documents through the LMS).

Before the class began, the instructor manually divided the students into 10 teams (namely *Team A* to *Team J*), each with 10 or 11 members. Three different types of teams were created: teams with Japanese students only, teams with international students only, and teams with a mixture of Japanese and international students.

Five applications were suggested to be discussed, each of them being covered by one, two or three teams, as follows.

Team A, Team B: language learning application
Team C, team D: medical records management system
Team E, Team F: low-budget oriented online shopping system
Team G, Team H, Team J: time/task management application
Team I: dating application for retired people

Purpose: create a requirements document for a given application.

The project consisted of 6 tasks, as described in Figure 2. At the start of the activity, each team chose two members who would act as stakeholders (*users: U*); the remaining members of the team would act as requirements analysts (*developers: D*). Thus, each team included two stakeholders and a maximum of 9 developers (depending on the number of members in the team). In theory, each team was made up of either 10 or 11 members; in practice, not all students were present on the day, thus some teams had fewer members.

The instructor believed that the inclusion of stakeholders in the group highlighted their crucial role when developing an application in the real world. According to the Guide to the Software Engineering Body of Knowledge [24], real-world software projects often suffer from two primary requirements-related problems: incompleteness (when stakeholder requirements are not revealed and properly communicated) and ambiguity (when requirements are communicated in a way that is open to multiple interpretations). Students need to

learn that the presence of stakeholders is essential in the requirements elicitation phase of developing a software system.

The resulting documents for each of the steps/tasks were saved in the LMS, using its "Project" function. Some teams chose to additionally create a shared google document and place the link in the project location.

Task I: Ds write a questionnaire for the two stakeholders
NB: Ds and Us do not communicate directly)

Task II: Us answer the questionnaire

Task III: Ds discuss with Ds from the other team(s) working on the same application
- Typical questions discussed include: *"What kind of questions did you include?"*, *"How can we find out what our application should do?"* etc.

Task IV: Ds must think of the following question. *Which is more efficient: a pre-defined questionnaire (you had time to think about it) OR interview (you can ask "live" questions?)*

Task V: Ds create a requirement document in free format (suggestions: use "shall", "should", possibly, also, "unclear" statements)

Task VI: Us give brief feedback on the requirements document - what they disagree with, what "pleasantly" surprises them, what is missing, based on their answers to the questionnaire prepared by the respective Ds

Figure 2. Description of tasks to be achieved during the class project

The students were seated in a large capacity classroom, which is organized in three sections of 3-person desks. Unfortunately, the desks and chairs cannot be moved or turned around, sometimes making it difficult for students to hold discussions in groups (they would have to turn their body to be able to speak to the colleagues seated behind). Figure 3 illustrates the seating of the students and a common manner of occupying the desks.

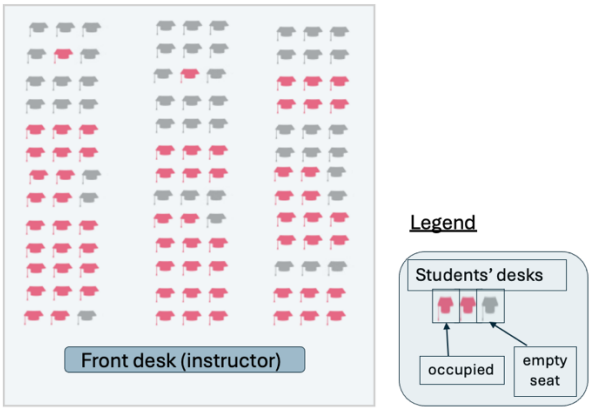

Figure 3. Visualization of clasroom seating

After the class ended, the students had to respond to a series of questions posed by the instructor, in the form of an assignment (submitted through the LMS). Importantly, the students had time until the end of the day to respond – thus allowing several hours to reflect upon the day's experience.

It should be noted that the instructor did not indicate whether the assignment was mandatory or linked to the final course grade. Her intention was to encourage students to share and reflect on their experiences only if they felt they had meaningful insights to contribute. In her view, making the task compulsory might have led some students to submit responses merely for the sake of compliance, rather than genuine engagement.

Feedback was received from 62 students (out of the ~70 students present on the day of the activity.) The requirements for the assignment are included below; the description also invited students to provide additional comments or suggestions, if they wished to do so.

Regarding today's activity, please let me know your thoughts (anything is fine - I am truly interested in your opinions!).

Part I. As a software developer
- Was there a leader among developers?
- Did everyone participate?
- Which do you think is "harder": to be the U ("stakeholder") or the D ("developer")?
- Is it better to have "smaller" or "larger" development teams?

Part II. As a class participant
- Did you enjoy the activity? (Please be honest, it is important for possible future activities!)
- Was it difficult to communicate with your colleagues?
- What was "the best" part today? What was "the worst" part today?
- Would you prefer your group members to speak the same language as you? (Is English ok for everyone?)

The comments submitted by the students highlighted various important and equally interesting issues. In the following, we shall focus on the answers given by students from a class participant point of view, i.e., Part II of the assignment.

The first question asked whether students enjoyed the activity; the majority of students declared that they did. We acknowledge that, even with the instructor's insistence on honesty, it is possible that social desirability bias was present - some students may have simply wanted to please the instructor. (Social desirability response bias is defined as a participant's tendency to over (under) report activities that are socially desirable (undesirable) [25].) Notably, since the answers were provided as assignments, they were not anonymous (they were added to each student's assignment portfolio) – an additional factor that may have influenced the students' responses.

Positive and encouraging responses included:
- *"It was fun and very different from what the other classes are doing."*
- *"YES, ideas shared by others are interesting especially if they are unexpected ones."*
- *"I enjoyed the activity. Compared to regular classes, I think this activity is a very interesting process and allows for a better grasp of the knowledge learned."*
- *"I enjoyed the activity. There were many diverse opinions, and they were stimulating."*
- *"Yes, can't wait for the next one."*
- *"I enjoyed it very much, especially interacting with others."*
- *"This activity provided me the chance to talk with other students and make new friends!"*
- *"It was great and I was able to talk to new people."*
- *"I enjoyed this activity. I am not good at speaking English, so I felt awkward to speak English and helplessness not to tell what I think well. However, I didn't felt such things this time. It is easy to communicate with my team. I was able to tell my opinion actively."*
- *"It was my first experience and I had a lot of fun." (translated from Japanese)*
- *"The class was very interesting because I never considered myself to be on the user side before."*

One notable answer was provided by a student who did not seem to enjoy the activity: "*My English is not very good. I can understand everyone's thoughts, but I have almost no practice in expressing myself. The better thing is that I can share my ideas by sharing documents. To be honest, it's not an enjoyment.".* The same student noted that it was difficult to communicate with their colleagues and that they would prefer for group members to speak the same language.

One other participant expressed their lack of satisfaction with the activity in the following manner: *"Everything felt arbitrary, and far from actual developing/consulting processes."* Unfortunately, they did not provide further clarifications, which would have helped the instructor understand the issue.

With regard to the preferred language of communication, most students seem to accept English as a common language. However, 6 students expressed their desire to use Japanese (it is worth remembering that 43 class participants are Japanese, although not all of them were present on the day of this activity). One student expressly stated that they would like to use English, even though their mother tongue is Japanese: *"I would prefer if we could speak in English, not put in a group where everyone speaks Japanese (I speak Japanese, but since it's an English class I'd prefer it that way)".* Another participant stated: *"Yes, I prefer to speak the same language. (Because Japanese student may not be able to follow the speed of discussion in English.)".*

When asked directly whether it was difficult to communicate with colleagues during the activity, only one student (out of the 62 who answered) specifically stated that it was (his answer was in Japanese). Although all the other participants considered that communication was not difficult, several comments highlighted the benefits of using one's own language for ease of communication and expressing own ideas; some examples are provided below.
- *"English is fine for everyone, but I think communicating in my native language would allow me to express my ideas more fluently."*
- *"[…] we have communicated in Japanese. Discussing in English should be harder than in Japanese."*
- *"I think communication is smoother and more efficient when group members speak the same language. English is fine, but Japanese is easier to understand."*
- *"I usually hesitate to speak English, but when I could speak the same language, I could be more active."*
- *"It is quite difficult to exchange opinions in English." (translated from Japanese)*
- *"As a Japanese, I think it is easier to have in-depth discussions about the content of the class if I am in a group with Japanese. (Although it does not seem to improve my English skills.)"*
- *"We want to speak the same language because it is easier to communicate."*
- *"I think communication would be smoother if we spoke the same language, but I think I'll improve more in English." (translated from Japanese)*

A particularly interesting comment was provided by one Japanese student: *"In the aspect of smoothly communication, it's better to have same language members. Inversely, in the aspect of diversity of opinions, it's better to have members with different languages."*. The student appears to be aware of the difficulties of speaking different languages (and, in the instructor's opinion, difficulties with Japanese vs. English), but also highly aware of the importance of diversity of opinions – different languages imply different cultures and thus "diverse opinions".

Various other students' opinions were elicited through this short assignment. One of these questions asked the students what the "best" and the "worst" part of the class project were. Two answers stood out among the students' responses.
- *"There were many diverse opinions, and they were stimulating. The best part of today's class was "when the opinions came together," and the worst part was "when no opinions were expressed.""*
- *"The best part is everyone working together collaboratively. The worst part is to write the report."*

As expected, cultural differences played an important part in the development of the class. For some students, as can be observed from the following comment, working with colleagues from the same cultural background is important: *"Grouping people of the same background together is the best part".* This observation aligns with the instructor's own empirical observations – she can often observe participants (Japanese students in particular), who, when offered a choice, decide to work in a mono cultural team (i.e., they make sure to be part of a Japanese only team). As shown by Rodriguez-Perez et al. [26], some of the problems that arise in diverse working teams "can be explained by the Similarity-Attraction

theory and the social categorization perspective". According to the Similarity-Attraction theory [26], individuals working in groups tend to prefer working with those who resemble them, while the social categorization perspective suggests that group members are more likely to trust, like and cooperate with similar others [27]. Classroom observations made by the instructor appeared to confirm the relevance of both perspectives.

In their responses, students mentioned other "best parts" of the activity, like leadership skills, working with the team to develop the questionnaire, collaborating with classmates and solving problems together, "communicating among us developers", "when everyone accepts each other's opinions". As for "worst" parts, one opinion stood out in particular: "*The "worst" is when I know we have to work as a team (shy)"*. This statement suggests that for some students, especially those who describe themselves as shy, the requirement to work in teams can be perceived as stressful or uncomfortable.

Through this "Project" activity, as performed in class, with the use of the LMS, students were faced with issues similar to those that arise during the requirements elicitation phase in the development of a real-life software product. Moreover, they could observe firsthand how cultural differences impact working in a team, as well as how they affect the overall process of software development.

B. *Game-Based Learning: Agile Paper Airplane Game*

One classical example of a classroom activity designed to simulate agile software development practices is the "Agile Paper Airplane Game [28]. This activity is particularly suitable for students who learn about agile methodologies (and Scrum [29]) for the first time, teaching them "the benefits of working in sprints, planning, retrospectives and teamwork" [28]. Students are divided into small teams and tasked with designing and building paper airplanes under iterative conditions. Instead of planning everything in advance, teams work in short "sprints," receiving feedback after each round on design improvements. Some important concepts are highlighted and experienced through this activity: continuous improvement, lean workflow and "Definition of Done" (defined as all the characteristics and standards a product increment needs to meet in order to be released [29]).

In our classroom, the students were divided into teams of 6-8 members. The teams were created based on the student seating. Before the class started, they had the opportunity to sit down in areas designated "Japanese language", thus belonging to groups in which everyone spoke Japanese. Only two groups of students were created in this manner. Incidentally, one team turned out to be made up of Chinese students only, thus making Chinese the language spoken in this group.

The activity started with the instructor explaining that the "goal" is to deliver paper airplanes that "meet customer expectations", i.e., fly a certain distance (the same one for all the teams). A leader was designated before any work started. Similarly to agile work, the teams worked in sprints – in our case, 3 sprints. In each iteration, the first minute was dedicated to planning; next, within three minutes, the teams had to fold and test as many paper airplanes as they could. After the three minutes of "development", they had one minute for retrospective – to discuss what could be improved towards the next iteration. One important rule stated that each team member was allowed to perform only one fold at a time, after which they had to pass the airplane under construction to the next team member. The third sprint came with a modified version of Definition of Done: the paper airplane's nose had to be blunt, and the required flying distance was increased.

Figure 4 shows three snapshots of the class taken during this activity, showing students folding airplanes and discussing during "retrospective" (included with the permission of the students).

Figure 4. Students participating in the agile paper airplane game

Notably, before each sprint, the teams were asked to estimate how many airplanes they would be able to build. "Definition of Done" plays an important role here: the airplanes must fly the designated distance, otherwise they cannot be counted as successful. After each sprint, the teams had the opportunity to refine their design – they made new decisions, under the guidance of the leader, during the one-minute retrospective. Each sprint focused on continuous improvement and responding to requirements. Teamwork and communication played a crucial role in this activity – aspects clearly noticed by the students, as could be seen from the comments they submitted after class.

When it comes to estimating how many "correct" airplanes their team would be able to build, the 3 sprints proved the concept of continuous improvement: as they advanced to the next sprint, the teams managed not only to estimate better the number of airplanes, but also to build more such planes which fulfill the Definition of Done characteristics. Figure 5 shows examples of the teams' estimations after each of the 3 sprints, along with the actual numbers of planes created, including "correct" ones. (The figure shows screenshots of the blackboard, as they were taken in class by the instructor.)

As can be observed, in the first sprint, the teams had no idea how difficult/easy it would be to build as many planes as possible within the 3-minute timeframe. Team 6, for instance, estimated that they would be able to build 150 planes (they ended up building 13, out of which only 7 fulfilled the DoD). In subsequent sprints, Team 6 greatly improved their estimations, as well as their results. In sprint 2, they estimated 10 planes and built 14 (with 12 "correct" ones), whereas in sprint 3, they estimated 15 and built exactly 15 "correct" ones (out of the 16 planes in total).

Figure 5. Estimates and actual figures for number of "correct" planes in the agile paper airplane game (7 teams, 3 sprints)

After this activity, in the usual style, the students were asked to provide their impressions. Again, they had time until the end of the day to submit their comments through the LMS, as a non-mandatory assignment. A number of 68 students provided feedback to the assignment described below.

Please let me know your thoughts regarding how things went today - any thoughts!
For example:
- Did you enjoy the activity?
- How was the communication with your colleagues?
- What was "the best" part today? What was "the worst" part today?
- Would you prefer your group members to speak the same language as you?
(Is English ok for everyone? etc.)

#Please include any comment/suggestion you think is important.

The students expressed their appreciation for this activity in various forms, as seen in the examples below.

- "I enjoyed the activity. I got the feelings of scrum participant. It was a good activity to easily experience the scrum development."
- "This was the best activity so far. Probably because it was so educational."
- "Thank you very much for this kind of activity; it improve[s] the understanding of real world work."
- "Today's activity was very enjoyable. It was an innovative way to experience agile development."
- "Really appreciate for the wonderful activity!"
- "Overall, this was the best activity so far, because it was educational and I learned more than I did from books and lectures (I'm sorry)."

During the informal discussions held by the instructor after class, she heard numerous opinions which classified this activity as the most enjoyable of the whole course. The game aspects which underpin game-based-learning proved to be very useful indeed: students had a taste of agile environments through a fun, game-like class activity.

Through this activity, the students became aware of some of the benefits of working in sprints, continuous improvement, retrospectives, as well as teamwork and the importance of communication and leadership.

For instance, one participant noted: *"I think the iterative improvement process is the most interesting"*. To go one step further, two comments regarding leadership stood out: *"These activities are good to find natural leaders"* and *"The leader decided the big folds and the order, which made me realize that the leader's presence was very important"*. Moreover, students recognized that managing people was another aspect which was illustrated through the activity: *"It was enlightening to learn about various aspects […]. Equally valuable was the insight into people management and how motivating team members plays a pivotal role."*.

When it comes to teamwork and communication, many students emphasized the usefulness of the activity in this respect, as shown in their comments, some of which are included below.
- "<The best> [part] is the group communication made me feel good, and I really like such kind of activity."
- "Communication was smooth for the most part. Everyone was eager to collaborate and share their ideas, which made the activity run smoothly."
- "The communication with my colleagues was effective and collaborative."
- "The best part today is the overall collaboration and teamwork were very enjoyable. It was great to see everyone working together and sharing ideas."
- "The process of folding airplanes is very interesting; everyone worked together in unity, which was great. There were no bad parts."
- "The communication between team members is very efficient, everyone expressed their ideas well, and everyone is very friendly."

In terms of language, which could be viewed either as a barrier or as a bridge to communication, several students

emphasized the importance of a common language (i.e., Japanese, for the local Japanese students), in order to communicate ideas clearly and effectively; the following two comments illustrate this idea.

- *"While English is fine and everyone can communicate, I think speaking in my native language would allow me to express my ideas more fluently and clearly."*
- *"Since Japanese is my first language, it would be easier for me if the group members could speak Japanese as much as possible. ".*

In one other instance, one of the (usually very active) students noted: *"In previous assignment I wrote I want to make a group with English speaker. I joined the English speaker group then, but I can't speak as fast as foreigners and they are so active, so I cannot act actively. It was more difficult than I expected. But, it doesn't mean I should do activity in same language group. I enjoyed this activity!"*. This student's reflection highlights both the challenges and benefits of participating in a multicultural group. After joining an English-speaking group, he found it difficult to keep up with the pace and high level of participation of more fluent members. Despite these difficulties, he recognized the value of the experience and, ultimately, found it enjoyable.

At the same time, students recognize that, more than the language, willingness to communicate is essential for an effective group work: *"I prefer my group members to speak same language of course because we'll get less communication mistake for language problem. However more important thing is not rather same language but rather will to communicate each other. With same language, we cannot communicate with unmotivated silent people."*

Some groups, made entirely of participants sharing a native language, performed very well and underlined the easiness of communication, as shown in the following comment: *"Since almost everyone in our group could say* [sic] *Chinese, I think our communication was effective and concise but enough for our work. After discussing our group work, we even had time to talk with each other."*. Another observation provided by a Chinese speaker (who can also communicate in Japanese) is relevant: *"As a Chinese student collaborating with Japanese peers, we used both Japanese and English to complement each other's understanding. This bilingual approach enhanced our teamwork and ensured that everyone was on the same page, which was critical for the success of the activity".*

Similarly to the cooperative activity described in the previous sub-section, although teamwork is central to software engineering, not all students view it positively. Shy or less confident students may experience group activities as a source of anxiety rather than engagement, especially when they involve colleagues that they do not know well, as illustrated by the following comment: *"It was an excellent activity. At first I felt a little shy about talking and working together with strangers, but after doing the actual works, I think we knew each other more and became more harmonious.".*

In conclusion, as one student summarized, this activity contributed to improving the students' communication and collaboration skills, providing valuable experience for their future: "*Overall, this activity not only enhanced our teamwork and communication skills but also gave me a deep understanding of the importance of process optimization and continuous improvement in scientific research and engineering practice. As a graduate student in science, I will apply these insights to my future studies and research, striving to improve my comprehensive abilities."*

V. Discussion and Lessons Learned

Teaching this introductory software engineering course in a multi-cultural classroom provided an invaluable experience, to both the students and the instructor. This section will offer further considerations regarding team formation, general course perceptions, as well as key lessons learned through the activities described in Section IV.

A. Team Formation

While conducting the class activities, the students found themselves in different settings, in the form of different kinds of teams - a different one each time. The process of team formation can "significantly impact the learning process, the social behavior of team members, and the team's overall performance" [23]. This concept has long been studied in social sciences areas, like resource management, sociology and psychology. Forming a capable team is significant for all kinds of organizations, businesses, sports, etc. ([30] [31]), just like it is important during the process of education, when students work in teams. However, in software engineering, studies on team formation are still relatively limited, compared to studies focused on technology and process-related aspects [32].

As explained earlier, three different types of teams were created throughout the course: teams with Japanese students only, teams with international students only, and teams with a mixture of Japanese and international students. The instructor chose different team formation styles for the purpose of understanding the best setting, the one which allows the easiest communication between students.

As student feedback showed, there is no perfect solution; however, the different arrangements offered students different experiences. Some Japanese speakers prefer mono-cultural groups, whereas other prefer to be challenged to speak English or to be part of groups with people from different cultures. The same is true for the international students: some of them find communication with other international students easiest, whereas others are very eager to make Japanese friends. Even when students had the choice to create their own groups, without the instructor's interference, various types of groups were formed, based on the same principles as above. Further work is needed to identify the best method of team formation in collaborative projects, in order to yield the best results in a multicultural software engineering class.

B. Student Course Evaluation

As explained in Section III, at the end of the course, as per university rules, the students are required to provide a course evaluation, based on a university-prepared questionnaire. Their answers are provided anonymously, following a Likert scale of 1 to 5, corresponding to "strongly agree", "agree", "neutral", "disagree" and "strongly disagree", respectively. As can be observed from the partially summarized results in Table II, with only a few exceptions of respondents choosing "neutral", all the participants selected either "strongly agree" or "agree" with the 5 statements. It is noteworthy that participants reported an increased interest in the subject following completion of the course. They mostly believed that the ways in which the instructor explained and planned the class contents were suitable for the course, and that there were sufficient opportunities to ask questions during class. Finally, their responses showed that the students were satisfied with the course: 19 chose "strongly agree", 11 chose "agree" and 1 chose "neutral".

TABLE II. RESULTS OF END OF COURSE QUESTIONNAIRE (1: STRONGLY AGREE; 2: AGREE; 3: NEUTRAL; 4: DISAGREE; 5: STRONGLY DISAGREE)

	1	2	3	4	5
"The instructions were well prepared for the course."	22	9	0	0	0
"The ways the instructor explained and planned the class contents were suitable for the course."	23	8	0	0	0
"Attending this course, I developed a stronger interest in the field of study related to this subject than before."	16	13	2	0	0
"Overall, I am satisfied with this course."	19	11	1	0	0
"You were given sufficient opportunities for asking questions to the instructor(s)."	24	5	2	0	0

The university-prepared course evaluation questionnaire allowed the provision of free comments, as well. One of them simply mentioned: *"I really like the group work and interacting with other students"*. In another comment, one student noted: *"This interactive opinion exchange class on software engineering principles provided a good opportunity for discussion of essential topics. It is well-designed. […] I was very happy to understand many different opinions!"* Notably, this student also included ideas for improving the course, by suggesting specific additional topics to be covered in the future (like SOLID design principles in C# [33]).

As one comment illustrates, at least some of the students were highly satisfied with the course: *"I am particularly grateful to [professor] for her dynamic teaching approach, which made complex concepts accessible and engaging. Her ability to convey intricate software engineering principles in a clear and practical manner significantly enhanced my understanding and interest in the subject. I have no suggestions for improvement at this time, as the course met all my educational needs. Thank you, professor […], for a truly enriching learning experience."*

Another set of data was obtained through the questionnaire prepared by the instructor at the end of the course. In this questionnaire, also anonymous, the participants were asked several questions; some of them elicited their opinions regarding multicultural classrooms, in terms of advantages/disadvantages of such an environment. Two more questions addressed the issue of whether cultural differences affect communication with their teachers, on one hand, and their student colleagues, on the other hand. The answers to these two questions are aggregated in Figure 6. As can be observed, about one third of the students believe that, in a multicultural classroom, communication is "definitely" affected or affected "most of the time" (a total of 36.67% students chose one of these two options, in both cases – with teachers and with colleagues). "Definitely not" was chosen by almost a quarter of the students (23.33%), for both situations of communication.

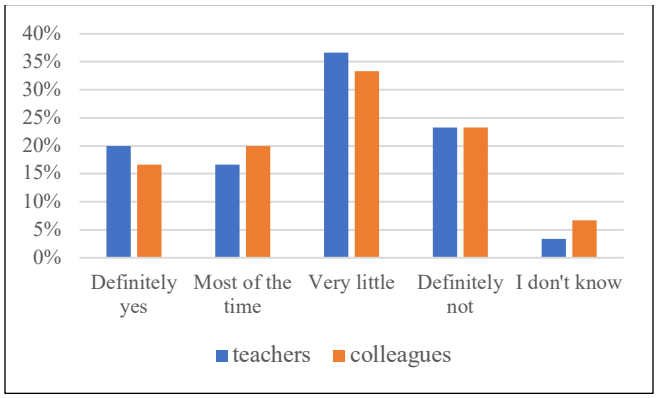

Figure 6. Answers to question "In your experience, do cultural differences affect communication/interaction with your teachers/student colleagues"?

C. Key Lessons

The key lessons drawn from our experience, as presented in this paper, are the following.

a) The software engineering classroom can provide an environment for students to experience developing an actual software product. Instructors must provide practical experience as much as possible, not only theoretical concepts.

b) Collaborative learning is especially useful in a software engineering classroom because it mirrors real-world team-based development, enabling students to practice technical problem-solving alongside essential communication and teamwork skills.

c) Games make learning fun and practical; game-based learning is valuable in a software engineering classroom because it increases motivation and engagement, while allowing students to apply theoretical concepts through interactive practice.

d) Multicultural classrooms provide students with an experience that reflects global work settings and offers a glimpse into software engineering practices across cultures.

e) The need for effective strategies to engage students in the classroom cannot be overstated. Fulfilling course objectives and achieving student satisfaction are closely tied to engagement, as higher levels of involvement typically translate into greater effort and commitment to learning.

VI. Conclusions and Future Work

Based on the experience of teaching a graduate software engineering course to a multicultural group of students, this paper highlighted ways to achieve student engagement through collaborative activities and games. Collaborative learning is an excellent way to prepare students for real-world, team-based software development, whereas game-based learning boosts motivation and helps students apply theory through interactive practice. Our study showed that group work in multicultural classrooms not only exposes students to linguistic and cultural differences but also helps them recognize the importance of stepping outside their comfort zone for personal and professional growth.

Future work will focus on evaluating the extent to which course objectives are met through the proposed strategies, and on identifying the most effective approaches for optimizing cooperative and game-based learning, particularly in multicultural settings, with attention to overcoming language and cultural barriers through suitable team formation. Moreover,. artificial intelligence can be considered to provide useful tools to overcome the language barriers in class.

References

[1] S. Vasilache, "Working together: Class activities as cross-cultural bridges in software engineering teaching," in *Proc. ICSEA 2024*.

[2] The Joint Task Force on Computer Science Curricula, "Computer science curricula 2023," [Online]. Available: https://ieeecs-media.computer.org/media/education/reports/CS2023.pdf. [Accessed: Nov. 30, 2025].

[3] S. Vasilache, "Bringing interactive instruction to the software engineering classroom: A multicultural group case study," in *Proc. IEEE Global Engineering Education Conf. (EDUCON)*, 2025, pp. 1–5.

[4] E. Lopez-Fernandez, P. Tovar, P. P. Alarcon, and F. Ortega, "Motivation of computer science engineering students: Analysis and recommendations," in *Proc. 2019 Frontiers in Education Conf. (FIE 2019)*, 2019.

[5] D. López-Fernández, A. Gordillo, P. P. Alarcón, and E. Tovar, "Comparing traditional teaching and game-based learning using teacher-authored games on computer science education," unpublished.

[6] M. A. Alsubaie, "Examples of current issues in the multicultural classroom," *J. Educ. Pract.*, vol. 6, no. 10, pp. 86–89, 2015.G. G. Borges and R. C. G. de Souza, "Skills development for software engineers: Systematic literature review," *Inf. Softw. Technol.*, vol. 168, p. 107395, 2024.

[7] M. Malcon-Cervera and C. Montaudon-Tomas, "Multicultural classrooms: Advantages for foreign and local students. A comparative study," in *Proc. EDULEARN17*, IATED, 2017, pp. 6477–6485.

[8] S. Vasilache, "Bringing interactive instruction to the software engineering classroom: A multicultural group case study," in *Proc. IEEE Global Engineering Education Conf. (EDUCON)*, 2025, pp. 1–5.

[9] M. Laal and M. Laal, "Collaborative learning: What is it?," *Procedia – Social Behav. Sci.*, vol. 31, pp. 491–495, 2012.

[10] M. Laal and S. M. Ghodsi, "Benefits of collaborative learning," *Procedia – Social Behav. Sci.*, vol. 31, pp. 486–490, 2012.

[11] "Collaborative learning vs. cooperative learning in the classroom," Promethean World, [Online]. Available: https://www.prometheanworld.com/resource-center/blogs/collaborative-vs-cooperative-learning/. [Accessed: Nov. 30, 2025].

[12] I. Garcia, C. Pacheco, F. Méndez, and J. A. Calvo-Manzano, "The effects of game-based learning in the acquisition of 'soft skills' on undergraduate software engineering courses: A systematic literature review," *Comput. Appl. Eng. Educ.*, vol. 28, no. 5, pp. 1327–1354, 2020.

[13] M. J. Sousa and Á. Rocha, "Game-based learning contexts for soft skills development," in *Proc. World Conf. Inf. Syst. Technol.*, Cham: Springer, 2017, pp. 931–940.

[14] B. S. Tan and K. S. Chong, "Unlocking the potential of game-based learning for soft skills development: A comprehensive review," *J. ICT Educ.*, vol. 10, no. 2, pp. 29–54, 2023.

[15] J. A. Medina-Merodio, A. Castillo-Martinez, R. Barchino, R. Estriegana, and R. Robina-Ramírez, "Factors influencing the acquisition of soft skills in a collaborative learning environment supported by game-based application," *IEEE Access*, 2024.

[16] J. Martí-Parreño, A. Galbis-Córdova, and M. J. Miquel-Romero, "Students' attitude towards the use of educational video games to develop competencies," *Comput. Hum. Behav.*, vol. 81, pp. 366–377, 2019.

[17] E. O'Farrell, M. Yilmaz, U. Gulec, and P. Clarke, "Playsafe: Results from a virtual reality study using digital game-based learning for safe agile software development," in *Proc. Eur. Conf. Softw. Process Improvement*, Cham: Springer, 2021, pp. 695–707.

[18] "The scaled agile framework (SAFe)," [Online]. Available: https://framework.scaledagile.com/#big-picture. [Accessed: Nov. 30, 2025].

[19] S. Beecham, T. Clear, J. Barr, M. Daniels, M. Oudshoorn, and J. Noll, "Preparing tomorrow's software engineers for work in a global environment," *IEEE Softw.*, vol. 34, no. 1, pp. 9–12, 2017.

[20] P. Schmiedmayer et al., "Global software engineering in a global classroom," in *Proc. ACM/IEEE 44th Int. Conf. Softw. Eng.: Softw. Eng. Educ. Training (ICSE-SEET)*, 2022, pp. 113–121.

[21] S. Vasilache, "A taste of distributed work environments: Emergency remote teaching and global software engineering," in *HCI Int. 2021 – Posters*, C. Stephanidis, C. Antona, and S. Ntoa, Eds. Cham: Springer, 2021, pp. 624–628.

[22] "Manaba," [Online]. Available: https://manaba.jp/products/. [Accessed: Nov. 30, 2025].

[23] J. Vilela, S. C. dos Santos, and D. Maia, "Impact of team formation type on students' performance in PBL-based software engineering education," in *Proc. CSEDU (2)*, 2024, pp. 327–338.

[24] IEEE Computer Society, "Software engineering body of knowledge," [Online]. Available: https://www.computer.org/education/bodies-of-knowledge/software-engineering. [Accessed: Nov. 30, 2025].

[25] A. M. O'Donnell and C. E. Hmelo-Silver, "Introduction: What is collaborative learning? An overview," in *The Int. Handbook of Collaborative Learning*, 2013, pp. 1–15.

[26] G. Rodríguez-Pérez, R. Nadri, and M. Nagappan, "Perceived diversity in software engineering: A systematic literature review," *Empirical Softw. Eng.*, vol. 26, no. 5, p. 102, 2021.

[27] A. C. Homan, D. Van Knippenberg, G. A. Van Kleef, and C. K. De Dreu, "Bridging faultlines by valuing diversity: Diversity beliefs, information elaboration, and performance in diverse work groups," *J. Appl. Psychol.*, vol. 92, no. 5, p. 1189, 2007.

[28] B. Willmott, "The agile paper airplane game," [Online]. Available: https://www.ppm.academy/post/the-agile-paper-airplane-game. [Accessed: Nov. 30, 2025].

[29] "Scrum," [Online]. Available: https://www.scrum.org. [Accessed: Nov. 30, 2025].

[30] P. Zainal, D. Razali, and Z. Mansor, "Team formation for agile software development: a review," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 10, no. 2, pp. 555–561, 2020.

[31] D. Strnad and N. Guid, "A fuzzy-genetic decision support system for project team formation," *Appl. Soft Comput.*, vol. 10, no. 4, pp. 1178–1187, 2010.

[32] M. I. Yilmaz, R. V. O'Connor, R. Colomo-Palacios, and P. Clarke, "An examination of personality traits and how they impact on software development teams," *Inf. Softw. Technol.*, vol. 86, pp. 101–122, 2017.

[33] N. C. Ramachandrappa, "SOLID design principles in software engineering," *Int. J. Comput. Trends Technol.*, vol. 72, no. 9, pp. 18–23, 2024.

[34] R. G. Tweed and D. R. Lehman, "Learning considered within a cultural context: Confucian and Socratic approaches," *Am. Psychol.*, vol. 57, no. 2, pp. 89–99, 2002.

[35] D. Byrne, *The Attraction Paradigm*, vol. 11. Cambridge, MA, USA: Academic Press, 1971.