Learning CS Subjects of Professional Software Development and Team Projects

Yu Chyi Chan
School of Computing Science, University of Glasgow
SIT@NYP Building
Singapore 567739
2609786C@student.gla.ac.uk

Chian Min Gan
School of Computing Science, University of Glasgow
SIT@NYP Building
Singapore 567739
2609779G@student.gla.ac.uk

Chun Yu Lim
School of Computing Science, University of Glasgow
SIT@NYP Building
Singapore 567739
2609794L@student.gla.ac.uk

Tien Hwa Tan
School of Computing Science, University of Glasgow
SIT@NYP Building
Singapore 567739
2609787T@student.gla.ac.uk

Qi Cao
School of Computing Science, University of Glasgow
Glasgow, Scotland, UK
qi.cao@glasgow.ac.uk
ORCID: 0000-0003-3243-5693

Chee Kiat Seow
School of Computing Science, University of Glasgow
Glasgow, Scotland, UK
CheeKiat.Seow@glasgow.ac.uk
ORCID: 0000-0002-6499-9410

***Abstract*—Professional Software Development (PSD) course is about the emerging profession of software engineering which involves developing, deploying, testing, and maintaining software. Currently, the Bachelor of Science with Honours in Computing Science (CS) joint degree programme of University of Glasgow (UofG) and Singapore Institute of Technology (SIT) carries out both PSD and Team Projects (TP) courses concurrently, where students are expected to apply the theory learnt from PSD to real-world projects in TP. TP is a practical project continuation from knowledge learnt in the PSD course. Both PSD and TP last two trimesters in parallel. This paper analyses the advantages and disadvantages of the current learning methods in PSD and TP. It is possible that there is still room for improvement in the current system. To further analyse the current learning system, a comparison of how the Software Engineering course is taught in other universities is also performed, from where ideas and methodology are proposed. The proposed methodology is analyzed and discussed from the suggestions or feedback of the CS students who have gone through both PSD and TP courses. The purposes are to improve the learning effectiveness of both PSD and TP courses.**

Keywords—Professional Software Development, Project-based Learning, Software Engineering, Software Development Life Cycle.

I. INTRODUCTION

Software Engineering or Professional Software Design (PSD) teaches students knowledge of software development life cycle (SDLC). SDLC knowledge includes topics of software process modelling, software architecture, continuous development, testing strategies, and software change management, etc. [1]. The knowledge of team organisation and software project management are also taught in this course [2]. As an interdisciplinary course, Software Engineering or PSD is offered to cohorts from multiple specialities, such as computing science (CS), information & communications technology, engineering, etc. Students with diverse background are expected to apply PSD skills or practices to design, develop, maintain, test, and evaluate computer software. The intangible nature of software products makes the learning of software engineering more complicated. This makes software engineering or PSD a hard subject to teach.

Students in the higher education are taught theory knowledge and process of SDLC [2]. But they are not exposed to real-world software projects often. Software industry has increasing demands for skilled professional software developers equipped with Software Engineering knowledge. It is necessary to train more software manpower from higher education [3]. What students learned in higher education have different focuses compared to the expectations from software industry companies' perspectives. Students do not have sufficient SDLC experience in real software industries and are often left unprepared for software projects in fast changing working environments. There is a learning gap from the theory knowledge into practical SDLC skills in higher education [4]. To close the learning gap, some universities including the Bachelor of Science with Honours in CS joint degree programme between University of Glasgow (UofG) and Singapore Institute of Technology (SIT) offer courses for both theory and practices [5]. In the curriculum of CS joint degree programme between UofG and SIT, both PSD course and Team Projects (TP) course are taught for two continuous trimesters concurrently. Team Projects course is for students adopting theory learnt from the PSD course and working in real software projects with real-world software companies. Students are grouped with 4 or 5 members for each software project. The combinations of the PSD and TP courses could provide students with the proper SDLC skill sets to anticipate ongoing changes of the SDLC technologies.

Although the combination of theory and practice courses is offered in the software engineering education, the learning effectiveness of such approach may be varied from different higher education settings. The motivation of this paper is to improve the learning effectiveness of both PSD and TP in the CS joint degree programme between UofG and SIT. As such, the juniors enrol in the Bachelor of Science with Honours in Computing Science degree can have a more effective learning system and benefit more from the modules as well as be more prepared to step out into the work industry with all necessary SDLC skills equipped.

According to the students' feedback, there are some limitations in the current setting of concurrent commencement of both PSD and TP. The research question for this paper would be: How effective is the current software engineering teaching method and how can it be improved?

The objective of this paper is to have a deep analysis of the teaching method and efficiency of PSD and TP to identify their current limitations and to propose ideas to improve the learning process. Additionally, a comparison of the teaching

method of Software Engineering in SIT with other universities is carried out in order to identify the areas of improvement to improve the learning effectiveness of both PSD and TP.

II. Related Work

This section discusses related works in the literature. The pedagogy and methodology used are studied and analysed.

A. Using Experimentation As a Teaching Tool

It is not always possible to have complete or real-world projects for students to work on due to resource constraints or course goals. Kuhrmann and Munch [6] reported to use experiments as a teaching tool in Software Engineering courses. The course was organised in three phases. The first phase (three weeks duration) were to provide the fundamentals required. In the second phase, students worked on the chosen topic and prepared small presentations as well as essays. The third phase wrapped up the course and reflected on the experiments. Experiments aid in identifying and analysing a problem to develop solutions, with quick feedback and learning allowed. Having a controlled environment allows students to experience risks and failure scenarios, as well as improve communication and collaboration skills. But the class size may be limited with this method.

Similar methods were also studied by other researchers, with an example presenting in [7].

B. Teaching Software Engineering without a Project Component

Most project-centric courses assume students already have software engineering or PSD knowledge and skills, despite some of the materials not being covered. The students are often expected to start the project before they learn the concepts, resulting in them learning these concepts after they have been working on the project for weeks, and often too late to apply them.

To prepare students for future projects courses, a course aimed at teaching software development principles and practices was designed and delivered without a project component for second-year undergraduate CS students [8]. The class consisted of three units and each unit ended with an examination. There were two lectures weekly and students had two weeks to complete the assignments.

In addition to learning programming, students learned how to share their codes effectively with their peers. They also learned basic design and usability principles, and how to use existing architectural and design patterns to maximise changeability in an iterative process.

After the course, students were prompted to provide feedback anonymously. The majority of them found that the assignments were useful and relevant. Surprisingly, students preferred on-demand access to the recorded lectures more than live lectures, even though they were not able to ask questions in the recorded lectures. However, some students commented that the assignments should be scaled down. Overall, the course was a successful one that met the learning objectives.

C. Teaching Software Engineering with Project Component

Some institutions designed a project-based software engineering course. A teaching experience report [9] describes that teaching theoretical concepts without linking them to practical applications may discourage the learning of students. Hence, they proposed project-based learning approach, combined with project management to create an environment that allowed students to deal with managers and real stakeholders. The purpose was to expose students to the realities of working on a software project in a corporate setting. The lectures were mixed with project lessons, allowing students to grasp theory before applying it to the project. The project was divided into four stages, each of which was linked to the subject covered in previous lectures. The four stages included requirements elicitation, planning & modelling, prototyping & systems integration, and presentation to stakeholders. The pedagogy was a success, and the students were enthused about it. However, some of the challenges encountered were: unbalanced groups of undergraduate students, varied times of each stage, and lack of planning and integration of the projects.

Majanoja and Vasankari [10] reflected their experience in organising capstone projects in a software engineering course. Based on the results, the technical aspects imposed significant challenges on the students. This was due to a lack of coding skills and resources to provide detailed technical guidance on all team-specific issues. Hence, it suggested strengthening students' technical skills before the project [10].

Recent advances in release management and collaboration workflows reduce the effort of students and instructors during delivery while also improving the quality of the deliverables. Bruegge *et al*. [11] presented Rugby, an agile process model based on Scrum that allows reacting to changing needs. Additionally, Tornado, a scenario-based design method that emphasises the use of informal models for client-student interaction was used to improve early communication. Students can deal with changing requirements, produce several releases, and get client feedback during the course using the combination of Rugby and Tornado.

Agile software development was identified as the major software engineering trend [12]. However, it is discovered that course materials and structure tended to deviate from real-life scenarios. It was suggested to combine Software Engineering courses with industry internships, so that students can learn agile practices and take on challenges in a professional setting.

D. Enhancing Learning using Reflexive Weekly Monitoring

The addition of Reflexive Weekly Monitoring (RWM) was explored to streamline the process of learning software engineering in conjunction with normal programming projects in an academic environment [13]. It was recommended for cases where novice developers run projects in teams while simultaneously taking other courses that may require the use of software engineering practices [14].

The RWM method was conceived to monitor development teams in a software engineering undergraduate course [13]. It used self-reflection and collaborative learning practices to help students be aware of their individual and team performance. It ensured that the module coordinators were aware of the progress of each student through weekly updates. The results obtained indicated that RWM was effective in enhancing the learning experience in the given scenario. 18 out of 32 teams in the study indicated that the monitoring sessions helped them increase their effectiveness, coordination, and sense of belonging to a team, but did not necessarily help their productivity.

E. Teaching using Integrated Active Learning Tools

Active learning is a technique of teaching that involves students actively engaging with course material through conversations, problem-solving, case studies, role plays, and other ways [15]. The use of active learning or teaching tools complements lectures and makes them more interesting for students while helping them in retaining knowledge.

To improve software engineering education by aligning it with academic and industry best practices, active learning teaching tools was developed in [16], consisting of class exercises, case studies, and case studies videos in partnership with the industry. 60 hours of software verification and validation (V&V) were created. The results showed that the software engineering knowledge taught through lectures was reinforced by this pedagogy.

A similar approach was also adopted at the University of Brasilia [17], that used a methodology of combining Problem Based Learning with Learning by Teaching. They believe that by having to teach another person the same concept, a person will learn more effectively. The students were required to produce questionnaires and videos to present their studies. The results showed that this learning environment was engaging and empowering students' learning. However, more time and effort were needed and some students were not interested.

Team-Based Learning (TBL) is one of the methodologies of active Learning. It is a form of structured small-group learning that emphasises students' preparation outside of class as well as the application of knowledge within a class [18]. An experience report [19] presented that adopting of TBL in software engineering courses demonstrated an increase in student engagement.

F. Teaching using Free Open Source Software

Educators have also been exploring using free open source projects as a teaching tool in software engineering courses. The term open source refers to software that has source codes that anyone can inspect, modify, and enhance [20]. Dorodchi *et al*. [21] have designed a course that focused on open-source, teamwork and modelling to teach the fundamentals of software programming. Students had the opportunities to work with open-source software to simulate working on an industry project and learn agile development. By using the open-source software, they have applied reverse engineering, software modelling, and project modification to include new features. Based on the results, students were dissatisfied with open source activities primarily because of the challenges associated with installation, configuration, and running on different operating systems. However, the impact of open source and teamwork were observed to be positive overall.

An experience report [22] showed that a collaboration with a large free open source software project could allow the students to gain benefits by incorporating principles of Project-Based Learning and Service Learning. Through the project, teachers could also teach several essential software engineering along the way. The experience was beneficial and invaluable for students, the teaching team as well as the open-source community.

Another group of educators from Towson University [23] introduced five learning activities using open source software as a teaching tool for software testing to provide students with real-world hands-on software testing knowledge and experience.

To learn more about the students' challenges, benefits, and attitudes toward working with open source software, Pinto *et al*. [24] conducted 21 semi-structured interviews with the students. Students reported that there was an improvement in their technical skills and self-confidence. Some of them found it extremely crucial for instructors to be involved with open source initiatives.

G. Teaching Software Engineering using Gamification

Gamification has been getting popular in educational settings [25]. Akpolat and Slany [26] from Graz University of Technology made use of gamification to enhance software engineering student engagement in an extreme programming course. The 50 student volunteers were randomly placed into five teams and were required to work on a pacific challenge about one of the extreme programming practices every week. The winning team of the week was rewarded with the challenge cup. Participants reported that they were learning more with gamification. There was also a discernible trend toward more intensive use of a practice that had been the focus of a weekly challenge.

Studies reported in [27] and [28] revealed a positive impact of gamification in teaching. Students felt more motivated and the approach enhanced their learning experiences.

III. Methodology

SDLC methodologies, the practices, and processes that are used by software developers teams as a direction to manoeuvre through the SDLC successfully. This section covers the current learning method and the proposed method of the PSD and TP courses in our CS joint degree programme.

A. Current Learning Method

In the current learning method, students are taught SDLC theory in the PSD course. In parallel, our university works with real customers from software industry for the real-world software projects in the TP course. It is to provide students the real working experience to interact with software companies. It allows students to practice while learning knowledge in the PSD course concurrently. In TP projects, team members are randomly assigned and tasked to bid for the projects that interests them. The learning process is segmented into six stages for the TP, mainly the Requirement Gathering day, four series of sprints with meeting the real customers monthly, and a Final Demonstration day.

As mentioned, the curriculum for PSD works in parallel with TP as well as many other modules. The course is designed to have students learning SDLC and working on a TP project in parallel. The learning outcomes including some key topics taught in the PSD course are shown as follows.

- Carry out selected software process models and Unified Modeling Language (UML) design for professional software development.
- Design and implement software architecture in SDLC.
- Perform software validation, verification, and testing for the developed software systems.
- Conduct software refactoring to enhance software functional and non-functional requirements.

- Ensure delivering high quality software under resource constraints by software project management and agile software development skills learnt in this course.

The learning journey for PSD includes weekly mini-quizzes, physical tutorials, two tests, a team project, and an examination. The weights of the assessment components are arranged next: weekly pop-up mini quizzes; 22% for each of the test 1 and test 2 as the closed-book tests physically conducted in the classroom settings; 20% for a team-based coursework project with four students per group; and 30% for the final exam in the classroom setting. The weekly pop-up mini quizzes are for the purpose of improving the student attainments in the course. The pop-up quizzes are conducted at random timing during each lecture, with multiple-choice question (MCQ) formats to test the students' understating on contents described in the same lecture. The pop-up quizzes are to examine if students follow the lecture discussions or not on the spot. But there is room to enhance the way of the pop-up quizzes been conducted, as there are currently no result statistics for each MCQ question on how many students made mistakes on which MCQ option.

1) Weekly Mini Quizzes

Students are required to participate in pop-up mini-quizzes during the lecture at random times throughout the lecture period. As believed, this is to ensure students pay attention during lectures, provide interactive sessions for the lecture and allow students to gain a deeper understanding. It contributes a total of 6% to the total mark for the assessment.

2) Tests

The current module consists of 2 tests, with one set before the recess week and another before the final examination. The first half of the assessment focuses more on the understanding and application of different UML Diagrams. Whereas the second half is on software management and code refactoring. Also, it contributes a total of 44% in total which is only slightly less than both examination and project report adds up. However, each of the tests weighs higher than the project report alone.

3) Project Report and Final Examination

The project report requires the students to undertake two project topics from the existing Team Projects. The teams are required to understand and apply what they have learned from the lecture. Teams are formed by students themselves, with the experiences and knowledge from the team project they are working on. The final section filled the remaining 50% of the entire assessment percentage.

4) Team Project Learning Method

The current learning method for TP ensures individuals are equipped with skills that are not just knowledge received from books. It shapes students to be self-disciplined and communicate as a team. It guides the team to meet deadlines and ensure timely deliverables while meeting customers' expectations.

B. Limitations Identified

As mentioned, the current method allows the students to learn and apply the SDLC knowledge concurrently. Customer collaboration allows students to have the opportunity to experience a real-world tradeoff in terms of meeting customers' expectations. Students will be able to respond independently when problems arise. The application of software knowledge enhances the understanding of students. The monthly progress demonstration also allows them to improve themselves in terms of communication skills.

The group assignment of PSD allows students to be software architects. They can redesign the current project using the knowledge learnt in PSD. This enables them to identify flaws in the current project planning, encouraging potential future improvement.

During the project closure phase, students are taught how to carry out code refactoring and testing, enhancing the quality and robustness of the software before handing it off to the customers. This invaluable experience is beneficial to future software projects.

However, there are various limitations to the current teaching method. Some of the limitations identified in this paper are knowledge not taught in time, different project difficulties and group formation.

As responded, some of the applications in software engineering do not meet the timeline of the TP. For instance, the UML Diagram was taught in PSD too late as teams were nearing the end of the TP when it was taught.

Secondly, the project scope of the various Team Projects had varying difficulties and different domains which makes it unfair for the teams that got the higher difficulty projects as they may not be equipped with the necessary skillset and require extra training themselves which results in inconsistent deliverables.

Moreover, group allocation methods have impact to learning experience [29]. Currently, the groupings of students in TP are randomly assigned before having to choose the topics. The general interest of the team might be different, leading to students being unmotivated in their work, negatively impacting their contribution to the project as well as the quality of software produced. The lack of motivation thus leads to them not contributing as much as other teammates that are passionate about the project which may result in an unfair workload on some team members therefore team morale is affected.

To add on, the change in team members for PSD changes the momentum as everyone is relatively new to their own TP and was required to grasp a deep understanding of other projects may not be ideal due to the huge range of projects available. Therefore, an additional effort has to be spent to understand another project which could prove to be a hurdle as the amount of time provided for the completion of the assignment is limited.

To resolve the limitations, the ideas proposed in this paper are knowledge to be taught earlier, choosing the TP projects before forming the groups, and making use of gamification to improve the overall experience.

C. Proposed Learning Method

Given the analysis of the advantages and limitations of the learning method, this paper proposes three methods to help improve the delivery of the courses of PSD and TP.

Firstly, the PSD theory should be taught beforehand to students before embarking on the TP project as mentioned in [8], especially knowledge of project planning and high-level design of the software. Having proper project planning is extremely essential to ensure the project starts right. Hence,

instead of running PSD and TP concurrently, fundamental software engineering knowledge should be delivered a few weeks before the start of the TP projects allowing students to have enough time to digest questions about uncertainty. Such a method would greatly improve the knowledge and enable a better understanding of the implementation of the project in different sprint phases.

Secondly, to address the current limitations, we propose that the team members should be able to choose the projects that they are interested in before they are grouped. This would ensure that all teammates would have the same interest and thus have higher motivation to do the project. Additionally, with similar interests, the team could be able to function better as everyone has a common goal. Furthermore, we would like to propose the grouping for PSD and TP to be the same, so that students do not need to spend more effort trying to understand another project in addition to their current one.

Thirdly, we suggest adding gamification elements to the teaching of the course as proposed in [26]-[28],[30]. A leaderboard could be set up to display the scores students obtained using Kahoot. We believe that this method can encourage student engagement in the lecture. Additionally, a healthy competition could be introduced between peers to further improve students' engagement and interaction with fast processing. As reported in [31], Kahoot enhanced student learning in the classroom, with the greatest effects reported on classroom dynamics, engagement, motivation, and an improved learning environment.

With these methods being proposed, they are presented and explained to the Year 2 students in the CS joint degree. Students are asked to provide their feedback according to their own personal learning experience in the PSD and TP courses. It is to measure how learners think is the proposed methods could address some of their concerns in the learning of these two courses. The survey questionnaires are conducted to validate the proposed methods next.

IV. Results and Analysis

According to our proposed methods to improve the learning effectiveness of the PSD and TP courses, survey questionnaires are conducted to gather the feedback and comments of students from the Bachelor of Science with Honours in CS joint degree programme of UofG and SIT.

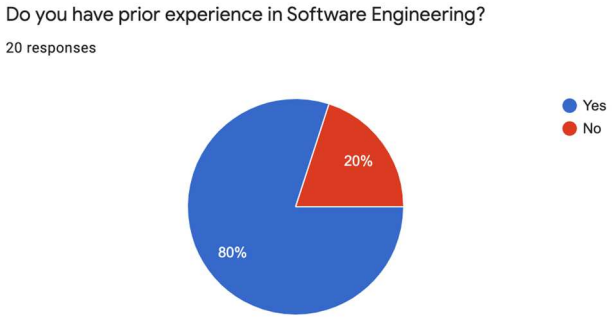

Fig. 1. Prior software engineering expertise.

20 participants out of 116 students are randomly invited from the cohort. We first ask if these participants have any prior experience in Software Engineering or PSD. Then according to the response, they rate their experience in the courses of PSD and TP, and how likely they will apply the knowledge they learnt in PSD in the future software development projects. They are also requested to share their feedback on the proposed methods.

Out of the 20 participants, 80% have prior experience in software engineering or PSD, while 20% do not have any experience before the PSD course as shown in Fig. 1. For those students with prior PSD knowledge, about 80% of them rate that they will highly likely apply the PSD knowledge in the future software projects.

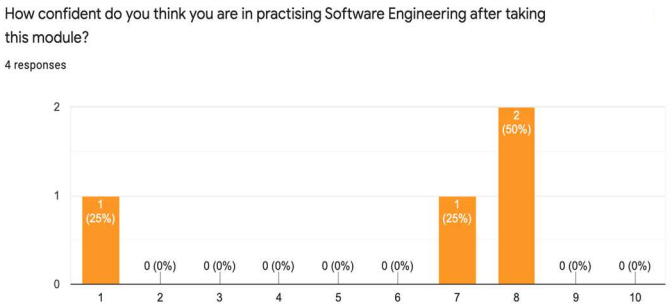

Fig. 2. How likely to apply knowledge for those without prior experience.

Whereas for participants with no software engineering knowledge, 75% of them vote that they will apply the PSD knowledge learnt into software projects as shown in Fig. 2. This shows that the current teaching method has a positive impact on learning.

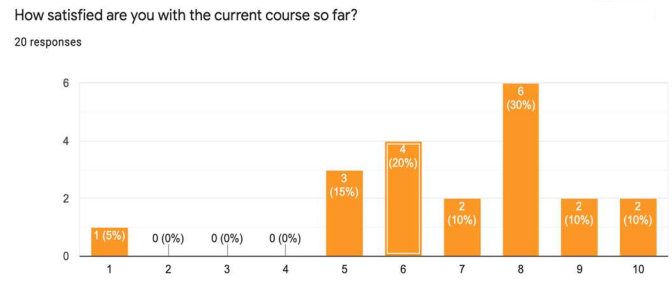

Fig. 3. Rating of satisfaction.

As shown in Fig. 3, most participants have a satisfactory experience with the course of PSD and TP. Despite the positive rating, participants have also feedback that there is room for improvement in the current teaching method. Some participants find that the concurrent teaching of both PSD and TP courses does not allow them to have time to fully grasp the knowledge due to a tight academic schedule and high commitment needed for TP, resulting in them producing subpar work for their TP. Additionally, some of the projects require them to explore a new field or technology. Extra efforts are needed to meet the expectations of the customers from the real software companies.

With regards to the teaching methods for PSD, with a scale above 7 as the benchmark, 70% find that the current method is effective, especially with the weekly lecture pop-up quizzes which increase their attention in class. Some participants have feedback that the continuous assessment enabled them to study the knowledge consistently without having to study everything at once before the final examination. There is also feedback such as the weekly physical tutorial sessions should be changed to online teaching as the time to travel to campus is longer than the length of the 1-hour tutorial sessions. They would prefer

online learning especially if there is no other lesson on the day of the tutorial.

65% of the participants have the opinion that the current delivery of the PSD course is related to TP. However, the other participants find that the contents taught are not quite related as their selected projects do not allow them to apply most of the knowledge learnt in PSD.

For the first proposed method on having the PSD theory be taught first, followed by the TP projects at later time, instead of the concurrent progress of both courses, most participants support it. It is observed the rating is 4.05 out of 5 in the Likert scale, as shown in Fig. 4.

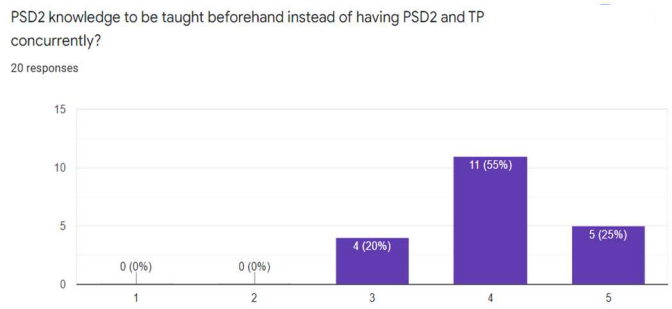

Fig. 4. Ratings for PSD knowledge to be taught before TP projects.

For the second proposed method on the group formation that allows students to choose their group members with the same interest on TP projects scopes, the responses of the participants are shown in Fig. 5, with rating at 4.05 out of 5 in the Likert scale.

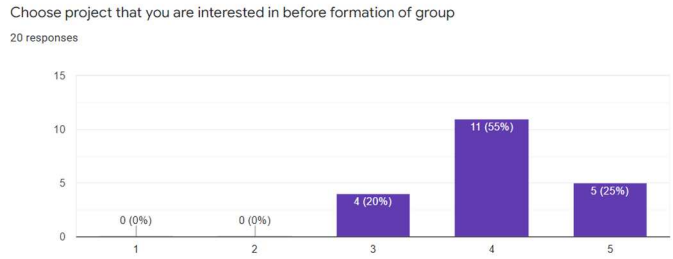

Fig. 5. Participants' ratings for group formation according to their interests.

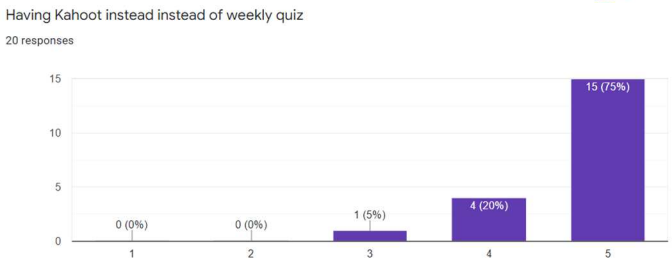

Fig. 6. Ratings for incorporating Kahoot method instead of a weekly quiz.

The responses of the participants on the third proposed method to add gamification elements in the PSD course are shown in Fig. 6. Based on the feedback from participants, it would be seen that the Kahoot method would be the most popular option to be implemented in future teaching methods. The participants have the opinion that through this method, they would pay more attention to the lesson in order to score better.

V. Conclusion

As the unique combinations of the PSD course and TP course offered in the joint Bachelor of Science with Honours in CS joint degree programme of UofG and SIT, it is an interesting topic to analyze and discuss the learning effectiveness from the viewpoint of students.

A few limitations of the current settings of PSD and TP courses have been identified in the paper. To address these limitations, three methods have been proposed to benefit the future cohorts when learning these two courses. It is to reiterate the importance of having to learn the PSD knowledge first, rather than teaching of both PSD and TP in parallel. As such, it serves to provide a better learning experience and ensure that the students are fully equipped with the respective needed PSD knowledge. As such, this proposed method would greatly enhance the quality of work delivered to the clients.

Providing some flexibility to students in the group formations of TP courses can also bring positive benefits in the learning. Using gamification component of Kahoot to replace weekly pop-up quizzes will also enhance learning enjoyment of students during the sessions. The assessment weightage for quizzes could be reduced to improve overall satisfaction without causing students to be demoralised even if they do badly in the first quiz.

With the survey findings, it is evident that the methods we propose and the feedback from the surveyees would bring great improvement to the curriculum and enhance the overall learning experiences of PSD and TP courses.

References

[1] I. Sommerville, *Software Engineering*, Pearson Publisher, 2018.

[2] R. Ferdiana, "Software engineering education learning process for professional developers," *Journal of E-Learning and Knowledge Society*, vol. 12, no. 2, 2016, doi: 10.20368/1971-8829/990.

[3] N. Assyne, H. Ghanbari, M. Pulkkinen, "The state of research on software engineering competencies: A systematic mapping study," *Journal of Systems and Software*, vol. 185, 2022, doi: 10.1016/j.jss.2021.111183.

[4] D. Oguz and K. Oguz, "Perspectives on the Gap Between the Software Industry and the Software Engineering Education," *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2936660.

[5] D. Hussain and L. Söderlindh, "Software engineering, bridging theory and practice in an agile learning environment," in *IEEE Global Engineering Education Conference*, 2022, pp. 541-546, doi: 10.1109/EDUCON52537.2022.9766486.

[6] M. Kuhrmann and J. Munch, "Enhancing Software Engineering Education Through Experimentation: An Experience Report", in *IEEE International Conference on Engineering, Technology and Innovation*, pp. 1-9, 2018, doi: 10.1109/ICE.2018.8436357.

[7] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Berlin, Heidelberg, 2012, doi: 10.1007/978-3-642-29044-2.

[8] P. McBurney and C. Murphy, "Experience of teaching a course on software engineering principles without a project," in *52nd ACM Technical Symposium on Computer Science Education*, 2021, pp. 122–128, doi: 10.1145/3408877.3432550.

[9] M.L. Fioravanti, B. Sena, L.N. Paschoal, *et al*., "Integrating Project Based Learning and Project Management for Software Engineering teaching," in *49th ACM Technical Symposium on Computer Science Education*, 2018, pp. 806–811, doi: 10.1145/3159450.3159599.

[10] A. Majanoja and T. Vasankari, "Reflections on teaching software engineering capstone course," in *10th International Conference on Computer Supported Education*, 2018, pp. 68–77.

[11] B. Bruegge, S. Krusche, and L. Alperowitz, "Software engineering project courses with industrial clients," *ACM Transactions on Computing Education*, vol. 15, no. 4, pp. 1–31, 2015.

[12] O. Cico, L. Jaccheri, A. Nguyen-Duc, and H. Zhang, “Exploring the intersection between software industry and Software Engineering Education - A systematic mapping of software engineering trends,” *Journal of Systems and Software*, vol. 172, 2021.

[13] M.R. Marques, S. Ochoa, M.C. Bastarrica, and F.J. Gutierrez, “Enhancing the Student Learning Experience in Software Engineering Project Courses,” *IEEE Transactions on Education*, vol. 61, no. 1, pp. 63-73, 2018, doi: 10.1109/TE.2017.2742989.

[14] L.B. Sherrell and S.G. Shiva, “Will earlier projects plus a disciplined process enforce SE principles throughout the CS curriculum?” in *27th International Conference on Software Engineering*, 2005, pp. 619-620, doi: 10.1109/ICSE.2005.1553615.

[15] L. Saunders and M. Wong, “Active Learning: Engaging People in the Learning Process,” in *Book: Instruction in Libraries and Information Centers: An Introduction*, 2020, doi: https://doi.org/10.21900/wd.12.

[16] S. Acharya, P. Manohar, P. Wu, A. Ansari, and W. Schilling, “Integrated active learning tools for enhanced pedagogy in a software engineering course,” in *ASEE Annual Conference and Exposition*, 2016, doi: 10.18260/p.24318.

[17] S. Antonio Andrade De Freitas, W. C. M. P. Silva and G. Marsicano, "Using an Active Learning Environment to Increase Students' Engagement," in *IEEE 29th International Conference on Software Engineering Education and Training*, 2016, pp. 232-236, doi: 10.1109/CSEET.2016.24.

[18] R. Mcdaniel, “Team-Based Learning,” Vanderbilt University, 2013.

[19] C. S. Ramos, R. A. Kosloski, E. Venson, *et al*., “TBL as an active learning-teaching methodology for software engineering courses,” in *XXXII Brazilian Symposium on Software Engineering*, 2018, pp. 289–297, doi: 10.1145/3266237.3266253.

[20] “What is open source?,” Accessed: Apr. 10, 2022. [Online]. Available: https://opensource.com/resources/what-open-source

[21] M. Dorodchi, E. Al-Hossami, M. Nagahisarchoghaei, R. S. Diwadkar and A. Benedict, "Teaching an Undergraduate Software Engineering Course using Active Learning and Open Source Projects," in *IEEE Frontiers in Education Conference*, 2019, pp. 1-5, doi: 10.1109/FIE43999.2019.9028517.

[22] A. Tafliovich, F. Estrada, and T. Caswell, “Teaching software engineering with free open source software development: An experience report,” in *Annual Hawaii International Conference on System Sciences*, 2019, pp. 7731–7741.

[23] L. Deng, J. Dehlinger and S. Chakraborty, "Teaching Software Testing with Free and Open Source Software," *IEEE International Conference on Software Testing, Verification and Validation Workshops*, 2020, pp. 412-418, doi: 10.1109/ICSTW50294.2020.00074.

[24] G. Pinto, C. Ferreira, C. Souza, I. Steinmacher and P. Meirelles, "Training Software Engineers Using Open-Source Software: The Students' Perspective," in *IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training*, 2019, pp. 147-157, doi: 10.1109/ICSE-SEET.2019.00024.

[25] “Gamification in education: What is it & how can you use it?,” True Education Partnerships. Accessed: Apr. 1, 2022. [Online]. Available: https://www.trueeducationpartnerships.com/schools/gamification-in-education/

[26] B. S. Akpolat and W. Slany, "Enhancing software engineering student team engagement in a high-intensity extreme programming course using gamification," in *IEEE 27th Conference on Software Engineering Education and Training*, 2014, pp. 149-153, doi: 10.1109/CSEET.2014.6816792.

[27] M. R. Souza, K. F. Constantino, L. F. Veado and E. M. L. Figueiredo, "Gamification in Software Engineering Education: An Empirical Study," in *IEEE 30th Conference on Software Engineering Education and Training*, 2017, pp. 276-284, doi: 10.1109/CSEET.2017.51.

[28] R. Malhotra, M. Massoudi and R. Jindal, "An Innovative Approach: Coupling Project-Based Learning and Game-Based Learning Approach in Teaching Software Engineering Course," in *IEEE International Conference on Technology, Engineering, Management for Societal impact using Marketing, Entrepreneurship and Talent*, 2020, pp. 1-5, doi: 10.1109/TEMSMET51618.2020.9557522.

[29] Q. Cao, L.H.I. Lim, V. Dale, N. Tasler, “Experiences in Python Programming Laboratory for Civil Engineering Students with Online Collaborative Programming Platform,” in *14th annual International Conference of Education, Research and Innovation*, 2021, doi: 10.21125/iceri.2021.1305.

[30] Q. Cao, B.T. Png, Y. Cai, Y. Cen, D. Xu, “Interactive Virtual Reality Game for Online Learning of Science Subject in Primary Schools,” in *IEEE International Conference on Engineering, Technology, and Education,* 2021, doi: 10.1109/TALE52509.2021.9678916.

[31] S. A. Licorish, H. E. Owen, B. Daniel, and J. L. George, “Students’ perception of Kahoot!’s influence on teaching and learning,” *Research and Practice in Technology Enhanced Learning*, vol. 13, no. 1, 2018.