
There may be differences between this version and the published version. You are advised to consult the published version if you wish to cite from it.

https://eprints.gla.ac.uk/287298/

Deposited on 13 December 2022
Sentiments Analysis and Feedback among Three Cohorts in Learning Software Engineering Modules

Yi Xuan Chia  
School of Computing Science, University of Glasgow  
SIT@NYP Building  
Singapore 567739  
2609721C@student.gla.ac.uk

Kai Heng Loh  
School of Computing Science, University of Glasgow  
SIT@NYP Building  
Singapore 567739  
2609718L@student.gla.ac.uk

Zhen Yu Brandon Ong  
School of Computing Science, University of Glasgow  
SIT@NYP Building  
Singapore 567739  
2609773O@student.gla.ac.uk

Jun Xian Lim  
School of Computing Science, University of Glasgow  
SIT@NYP Building  
Singapore 567739  
2609724L@student.gla.ac.uk

Jun Kai Ooi  
School of Computing Science, University of Glasgow  
SIT@NYP Building  
Singapore 567739  
2609711O@student.gla.ac.uk

Qi Cao  
ORCID: 0000-0003-3243-5693

Peter Chunyu Yau  
School of Computing Science, University of Glasgow  
Glasgow, Scotland, UK  
PeterCY.Yau@glasgow.ac.uk

Li Hong Idris Lim  
School of Engineering, University of Glasgow  
Glasgow, Scotland, UK  
LiHonIdris.Lim@glasgow.ac.uk

Abstract—Computing Science (CS) and software programming skillsets are widely used in many fields. The acquisition of software engineering skills is important and crucial to prepare students for the software industry. Learning of professional software development (PSD) knowledge involves various educational approaches. This study examines the feedback in students’ learning journey and preparedness for the industry across three years of the CS degree programme. The research is carried out to evaluate the current bottlenecks of learning in the PSD course. Surveys are conducted to collect feedback of 72 CS students from three different cohorts, namely freshmen, penultimate and final study year. Their sentiments are identified and analyzed using the quantitative and qualitative data gathered. Observed from the results obtained, most of the final year students have perceived attainment of favourable skills needed for the industry, after their internship experience where they applied their skillsets. A significant rise in positive sentiments from the students suggests that the PSD module taught is relevant and applicable for the software industry. However, the sentiments among the penultimate year students are different and show that there might be an issue with the current course delivery, which needs to be improved. In addition, this paper suggests possible improvement opportunities that could be explored by eliciting current challenges faced in the current syllabus. These include the specification variances for projects, timely introduction of topics, flexibility in marking rubrics, and introduction of in-demand practices.

Keywords—Computing Science, Software Engineering, Learner Differences, Sentiment Analysis, Learning Feedback

I. INTRODUCTION

Software engineering practices and methodologies are commonly taught in Institutes of Higher Learning (IHL). The main goal of teaching software engineering is ultimately to instil correct software engineering principles of functionality, reliability, usability, efficiency, maintainability, and portability [1]. These practices in the software development life cycle (SDLC) help prepare undergraduate students for the future industry work, where usage of such practices and knowledge is widespread. Similarly, in the Bachelor of Science (BSc.) in Computing Science (CS) offered jointly by Singapore Institute of Technology and University of Glasgow (SIT-UofG), software engineering education forms the foundational core of the bachelors’ degree in the syllabus. The key core modules where such SDLC principles and processes are introduced are in courses of Professional Software Development (PSD) and its practical counterpart course, Team Projects. These two courses are conducted in parallel in the penultimate year. The PSD course aims to provide students with the necessary knowledge and skills towards the SDLC and processes. The PSD course teaches various theoretical software engineering topics that will be applied in the Team Projects course. Gaining insights into processes such as software processes, project planning, team communication, software modelling, software architecture and continuous integration prepares students for the Team Projects course. This course involves industry clients from external software companies, which provide the real-life problem statements for the software development projects. Students are divided into groups with about 5 members per group.

Albeit the software engineering modules that undergraduate students have taken prior to their graduation, the software industry constantly shows dissatisfaction, with regards to the preparedness of the graduates. This is explicitly shown through extended training provided by the industry for graduates [2]. This iterates a potential problem of insufficient preparation of graduates for the industry, and opportunities for such areas can be discovered and improved. As such, more can be done to prepare undergraduates better in the software engineering domain.

This study aims to identify bottlenecks in the current courses and strives to discover the potential improvement opportunities for software engineering courses, including the PSD and Team
Projects, to prepare undergraduates for the software industry. In this paper, our study is performed according to the task structures as follow. We first discuss the current methodology to conduct both the PSD and Team Projects courses adopted in the curriculum of the joint CS degree programme of SIT-UofG. To evaluate and benchmark the student feedback towards on both PSD and Team Projects courses, survey questionnaires are performed by inviting students from three different CS cohorts. The sentiments analysis for the feedback of these three cohorts are performed next. With the benchmark results, our analysis is discussed to sense the potential improvement opportunities on the teaching methodology of both courses.

The remaining sections of this paper are organised as follows. Literature review is conducted in Section II. The proposed investigation methodology is presented in Section III. The survey studies are performed and results are described in Section IV. Identified opportunities for improvement are discussed in Section V. Conclusions are presented in Section VI.

II. RELATED WORK

A. Current Methodologies

The Agile and Scrum software development processes are commonly used nowadays, to educate students on software engineering or software development. There are various methodologies adopted by universities across the globe to teach the Agile development process in their syllabus. It can be observed that partnering with industrial experts [3], project-based learning [4] and communication-based learning [5] have appeared the most methods among studies conducted.

Multiple schools have been invited or partnered up with industrial experts to supplement education with conducting guest lectures for their students [3]. Coursework and assignments are greatly resembled with the real-life process of software development including estimations of work to be completed, conducting the projects following the SDLC closely, and peer reviews. Students are graded not only on the quality of the software, but also on their software development process by industrial experts and lecturers.

Many universities have also opted for project-based learning rather than having a one-way delivery type of education [4]. The focus of project-based learning requires them to understand the SDLC and put what they have learnt into practice. It usually consists of three steps. First, students are assigned a problem statement and a stakeholder to which students need to find a solution in certain amount of time. Next, students conduct problem analysis and requirement gathering before starting the development. Finally, they design a solution prototype and present a mock-up of their final product.

The key to communication-based learning is the integration of practising Agile development process while communicating effectively to teammates and clients [5]. In this methodology, students are given the assignments in which they need to propose a solution. To demonstrate their understanding of the software process, they are expected to apply the Agile manifesto in the projects. After which they need to make the presentations to clients and lecturers. One of the more interesting elements to note is that in this presentation, the presentation slides are set to scroll automatically at a time unknown to the student, so they have to react accordingly to the content displayed. Finally, students are asked to write a reflection essay and present it again.

In comparison, the current methodology used by the CS degree programme offered jointly by SIT-UofG encompasses a project-based, agile focused curriculum. It requires students to develop software programs for real-life clients from the software industry. In the curriculum, the PSD and Team Projects courses are developed as follows, one provides learning of SDLC theory knowledge and the other provides project based learning through practice. Both PSD and Team Projects courses are offered to CS students of the penultimate year in parallel for continuous two trimesters. Each course counts for 10 academic credits. The contact hours for the PSD course include two hours lectures and one hour tutorial every week. The Team Projects course include two hours contact hours and offline software project development every week. Software companies are invited to provide their real world software project requirements to the students in the Team Projects course, where each group of students works on a software project for one company. Within two trimesters, students need to complete the software projects and hand over the solutions to the clients, i.e., the software companies.

Students are assessed by their ability to satisfy the needs of the software company by developing software for them, based on the agile practices that they have learnt in their lecture. Students are also required to communicate frequently with the clients through regular project meetings or other mediums.

B. Advantages and Disadvantages of Current Methodologies

It is realised that most students right after graduation often do not possess the needed skills, abilities or even knowledge to perform in the industry [6]. They usually lack an understanding of software engineering concepts like SDLC, experience working with clients, software tools skills, problem-solving skills, project experience, as well as oral and written communications skills. This is often due to the knowledge gap between what students learn in class versus the actual requirements of the industry. For example, in a worst-case scenario, what students learnt might already be phased out by the time they enter the industry with the fast-changing world of technologies [7].

For the first methodology of industry partnerships, the advantages of partnering with industrial experts are apparent. By doing so, the students will be able to get involved in an actual industry projects gaining valuable experiences. At the same time, they fulfil most if not all requirements demanded by the industry, as mentioned in the previous paragraph. This approach is also the best way to learn practical knowledge such as working with a real client, teamwork and SDLC processes [8]. However, even though partnering with industrial partners may have the most benefits for the students as it provides the closest and most authentic experience working in the industry, it does have its disadvantages. One such disadvantage would be that students might feel stressed out by the need to deliver their client’s requirements while juggling their studies. As another disadvantage, students have little room for errors as they are working with an actual client. As such, there is a need to deliver working software at the end of the course, else risking clients being dissatisfied [9].
Next, for the methodology of project-based learning, the advantages include allowing students to experience the processes of a SDLC, invoking their problem-solving skills and experience working in a team to implement solutions [10]. As project-based learning usually employs made-up problems statements in the industry and does not involve actual industrial partners, this creates a safe space for students to make mistakes and learn from mistakes. It allows students to explore new ideas. At the same time, providing them with similar experiences of solving a real-world problem in the industry. On the other hand, the project being not a real industry project with actual industrial partners could also be a disadvantage. As the problem statement made up by the school for the project might be outdated with the actual requirements of the current industry. It leads to the students learning outdated and redundant knowledge that might not serve as valuable to students when they graduate.

For the methodology of communication-based learning, the advantage is that students can gain an in-depth theoretical understanding of software engineering concepts, such as the SDLC processes and the agile manifesto. Nevertheless, the shortfall of communication-based learning is that the student may still find it hard to apply the theoretical knowledge to real-world scenarios they face when they enter the industry. Most likely, theoretical knowledge does not translate well to practical knowledge. Hence, in this methodology, the students might find themselves lacking in practical experience when they graduate.

C. Experimental Educational Approaches

With rapidly changing technology and software in the current software industry, traditional software engineering education approaches might not be effective. As a result, there is a natural need to innovate and discover alternative teaching frameworks and approach.

Several IHLs have enhanced their software engineering modules to provide a more insightful learning experience and better prepare undergraduates for the employment in the software industry. For example, empirical studies of software engineering are conducted with undergraduate students, enabling them to learn state-of-the-art emerging technologies and frameworks [11]. This also translates to more industry relevance as such studies are conducted with an industrial goal. However, given the vast information available in the software industry, there is too little time to cover all of such information. Potentially resulting in lost education time and inadequate technological training.

To achieve a balance between emerging technologies and to not overlook the fundamental information, stable and in-demand practices are combined with project practices for an enhanced learning experience. One emerging practice that is set to grow is DevOps that is a set of practices combining software development (Dev) and operations (Ops). Such cultures are adopted alongside projects that undergraduates will be doing [12]. Continuous Integration and Continuous Delivery (CI/CD) is part of the toolset related to DevOps, which teaches the importance of static analysis, integration testing, unit testing, and code building [2]. Emerging tools like TeamCity, Jenkins, and CircleCI are introduced, preparing undergraduates with experience using widely adopted industry tools and services. Other technology enhanced learning methods such as cloud computing platforms and 3D visualization tool could also provide complementary ways to improve the learning efficiency [13]. Such educational approaches have resulted in better absorption of information and a better understanding overall.

Another alternative experimental approach to software engineering education is Challenged Based Learning (CBL), which aims to hasten the development of software engineers in specialised fields [14]. In this method, instead of planning out the module by lecturers, students are sent for specialised CBL training to better facilitate the process. The undergraduates also have to undergo stringent interviews and tests to ensure their competency and capability to approach such learning methodologies, streamlining the educational process.

III. OUR METHODOLOGY

For the assessment of the concurrent education approach of software engineering principles in the PSD and Team Projects courses in our joint CS degree programme of SIT-UoFG, quantitative and qualitative data gathering methods are employed in this paper. A mix of both qualitative and quantitative data are garnered through survey questions strategically crafted for three different CS student cohorts: first-year students, penultimate year students, and final year students. Leveraging on questions based on the Likert Scale and few open-ended questions, the sentiments and reflections of students from different cohorts are the primary data acquired. Surveying across the three cohorts provide good representation and comprehensive real-time insights of the programme [15]. Surveys are used as it is a cost-effective and efficient method for collecting information and feedback on student perceptions of the educational structure of the university. There is a total number of 72 CS students participating in the survey studies including 30 first-year students, 21 penultimate year students, and 21 final year students.

A. Surveys Gathering the Sentiment of Freshmen

As many of the freshmen enrol into IHL from different backgrounds, some with existing knowledge of software development, and others from non-CS disciplines of studies with less relevant knowledge. The target of the surveys for the freshmen is to assess their current sentiments and gauge their current understanding of software development based on previous knowledge. Information regarding their impressions and expectations of the software engineering course will be also collected.

B. Surveys Gathering Sentiment of Penultimate Year Students

To gather the latest data regarding the sentiments towards the current structure and educational methods of software engineering modules in the university, undergraduates in their penultimate year of study in the joint CS degree programme of SIT-UoFG will be surveyed. Surveying penultimate year students can potentially bring highly valuable insights as these undergraduates have just undergone the core software engineering courses, i.e., the PSD and Team Projects courses. They hold the latest sentiments and reflections on the current state of education in the University.
C. Surveys Gathering Sentiment of Final Year Students

For the final year students in the University, they have already completed most of the CS courses, including the software engineering models. Some of the final year students may already have working experience during their internship attachments with software industry companies. The survey questions will be on how the current curriculum has helped them with their internship work. In addition, the questions pertaining to which modules are the most helpful based on their field of work is also included.

IV. RESULTS AND DISCUSSIONS

In line with the methodology outlined in Section III, the surveys are conducted in this study to gather the sentiments of the three cohorts of freshmen, penultimate students, and final year undergraduates. The surveys are designed to be short and straightforward with minimum open-ended questions to prevent survey fatigue.

A. Freshmen’s Expectations and Impressions of the Module

The survey asks if the students have learnt similar software engineering modules in their prior study before enrolling into the University. One fascinating insight that we discover through the freshmen survey is that 66.7% of 30 respondents do not learn courses similar to PSD and Team Projects, as seen in Fig. 1.

![Fig. 1. Freshmen respondents for similar modules taken.](image)

Through further analysis of the responses, our research team discovers that many of these respondents have gone through similar modules in their previous educational institutions. Having not gone through these modules, sentiments of having these modules as part of their university syllabus are high, as shown in Fig. 2. It shows that most respondents have good expectations on learning the software engineering courses, as they perceive the importance of the software engineering courses.

With over 93.3% of respondents indicating a somewhat positive sentiment, it shows that many freshmen students are looking forward to taking these software engineering modules.

B. Expectations and Impressions of Penultimate Year Students

In the survey crafted for the penultimate year students, the students are asked to provide their feedback and comments on their recent experiences in the learning of the PSD and Team Projects modules in the previous trimester.

The feedback results collected provide some interesting insights. One such example is less than half of the respondents, 42.9% agree their initial expectations of the modules were met, as shown in Fig. 3. The results are very different from our initially expected higher ratio of “Yes” responses.

![Fig. 3. Expectation analysis of penultimate year students.](image)

Through the analysis, this might be because the respondents who chose “No” felt that the theories and concepts covered in the PSD course were not in-depth enough, or not well delivered enough to meet their learning expectations. Secondly, the survey has also shown most penultimate year students felt the courses did not help them grasp the concept of software engineering well, with higher respondents selecting ratings of 1, 2, 3 and 3, observed from Fig. 4. This is likely because both the PSD and Team Projects were conducted concurrently, resulting in a lack of time to apply theories and concepts learnt in the PSD to practical use in the Team Projects. Hence, the students might have felt the schedule was too rushed, affecting their grasp of the concept of software engineering.

![Fig. 4. Sentiments of module effectiveness by penultimate year responders.](image)
Respondents are asked to provide their comments on how the PSD and Team Projects courses can be improved. As shown in TABLE 1, there were various suggestions for improvements collected. However, through further analysis, a repetitive pattern to the suggestions was identified where almost every respondent suggested improvements for the arrangement of the Team Projects, such as balancing project scopes, project difficulties and number of group members. Students are grouped with about 5 members per group in one project group.

TABLE I. IMPROVEMENT SUGGESTIONS FROM RESPONDENTS

<table>
<thead>
<tr>
<th>What do you feel can be improved?</th>
<th>21 responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>The difficulty of team projects could be balanced better, some projects are very hard and some are very easy</td>
<td></td>
</tr>
<tr>
<td>The applicability of learnt knowledge in projects. Not all projects are viable for most of the knowledge learnt in PSD to be applied to.</td>
<td></td>
</tr>
<tr>
<td>The projects can be better and more equally distributed</td>
<td></td>
</tr>
<tr>
<td>The projects and the team groupings</td>
<td></td>
</tr>
<tr>
<td>more industry skills like DevOps and CI/CD, and more evenly spread out projects as well as group members</td>
<td></td>
</tr>
<tr>
<td>The content and the projects</td>
<td></td>
</tr>
<tr>
<td>The projects are unequal, and being able to find our own groups would be better</td>
<td></td>
</tr>
</tbody>
</table>

C. Final Year Students’ Impressions of the Module

To better compare the eventual benefits of the core software engineering courses that the University has taught, surveys are conducted to mainly assess the applicability of such modules, targeted at final year students who have working experience through their university industry internship attachments. They were asked how the software engineering courses learnt in the University are helpful to prepare them in the SDLC.

The survey received 21 respondents from the final year students, shown in Fig. 5. It is observed that about 81\% agree that the PSD and Team Projects courses have prepared them for acquiring skills of the software engineering.

![Fig. 5. Sentiments of module effectiveness by final year responders.](image)

It can be inferred that most of the final year students found that the PSD and Team Projects modules have enabled them to absorb software engineering concepts better. However, this is somewhat contradictory to the trends observed from the results obtained of the penultimate students shown in Fig. 4.

One rational assumption would be that the penultimate students have not had any software industry working experience, as they have not gone for their internship attachments yet. It results in them not finding the materials taught in the PSD and Team Projects relevant. On the other hand, final year students realise the relevance of the PSD and Team Projects modules when they work in their industry attachments. Similarly, respondents for the final year students have found that the skills learnt in the PSD and Team Projects are applicable and useful in their careers and the software industry, as shown in Fig. 6.

![Fig. 6. Final Year respondents on knowledge absorption.](image)

As technological advancements in the industry evolve quickly, school education needs to be enhanced with updated teaching materials, learning pedagogy, and technology enhanced learning approaches [16]. Given the current industry where there are many emerging technologies to be learnt [17], many respondents in the final year students still found the skills taught in the PSD and Team Projects courses to be applicable and useful. The majority of them found the knowledge learnt in these two modules to be relevant to their careers in the industry.

This might potentially be due to the knowledge learnt in these two courses contributing heavily to the foundational knowledge for most industry technologies and practices, such as software modelling, architecture, Unified Modeling Language (UML) diagrams, etc. [18]. It results in the contents taught being versatile and transferable in the software engineering and practices domain.

D. Overall Sentiment Analysis

A particular trend can be observed from the survey results for the respective target segments. The sentiments of freshmen in the CS cohorts are rather good towards taking the PSD and Team Projects modules. However, the sentiments of the penultimate year students are relatively more towards the negative side. In contrast, the final year students’ sentiments are contradictory to the penultimate year students, with most having good sentiments about the PSD and Team Projects modules. One possible rationale that the research team has derived is that the first-year students are really excited and looking forward to consuming more knowledge in introductory and foundational software engineering modules like PSD and Team Projects. Thus, such excitement and sentiments naturally lead to higher expectations.

The penultimate year students are the ones who have just completed their PSD and Team Projects modules. They might have found that the contents covered by these two modules do not meet their artificially high expectations, which were developed during their first years. Additionally, having just been involved in only one project in the Team Projects module, they might have a negative sentiment as they have not
experienced applying the PSD principles and methodologies taught in other projects yet.

However, the final year students have a significantly positive sentiment towards the PSD and Team Projects modules. The majority of the respondents state that the knowledge acquired in these modules is applied in their additional projects and industry attachments. This strongly supports and justifies the previous hypothesis of the sentiments of penultimate year students.

Overall, the sentiments across all students undertaking the CS joint degree in SIT-UofG are relatively positive, except for the penultimate year students. The analysis results suggest that there is still room for improvement on the teaching of the PSD and Team Projects course in the penultimate year. When the teaching contents of both courses were offered in the previous years, no students feedback was collected and analysed. The expectations of the penultimate year students were not placed into consideration previously. With the sentiments analysis being performed and benchmarked in this paper, it indicates the right directions for us to improve the teaching methods for both courses. In spite of that, there are points that strongly suggest our hypothesis of such sentiments, which strongly suggests that these sentiments would generally turn positive if these students get opportunities to be involved in more projects over time. Moving forward, in order to give students more opportunities to practice the SDLC knowledge into projects, we could add mini projects into the PSD course. Students can practice the SDLC knowledge in the mini projects first, before they join forces to develop the solutions for the Team Projects for the software companies. It could better help the penultimate year students turn into positive sentiments for both courses.

V. OPPORTUNITIES FOR IMPROVEMENT

From the results of the surveys conducted, these sentiments strongly suggest that there are opportunities for improvement, with regards to the PSD and Team Projects courses. This section will elaborate on some of the improvement opportunities that the research team suggests.

A. Project Variability in Team Projects

One of the pain points that some of the students have encountered in the Team Projects course is the variance of the different software development projects. Some software industry projects are centred around specific specialisation domains and niche areas, as compared to some that revolve around general software development. An example of such specialised software would be a project in the Team Projects course where Deep Learning and object detection are required compared to a more generalised software project that deals with web and full-stack development. The former project is in a completely foreign domain to some students, significantly hindering project development and applicability of learnt knowledge in the PSD course.

In addition, such variances in projects make it hard for some groups of students to apply what they have learnt in such projects. These projects may have strong dependencies on other systems or are only applicable when executed on a specific system. It makes certain introduced topics such as software testing, system integration, and continuous integration, etc., rather challenging as such practices require another similar staging environment.

Potential improvements could be made with regards to the balancing of such projects, which will significantly increase the applicability of applied learning from knowledge learnt from other modules.

B. Timely Introduction of Topics Along Concurrent Modules

To efficiently disseminate knowledge and facilitate the retention of such information, the teaching framework of Gagne’s Nine Events of Instruction suggests that learning activities should be strategically introduced to keep students involved [19]. Certain topics are found to be introduced relatively late into the software development of the Team Projects. It makes the application of such knowledge challenging or the need to apply refactoring to some initial assumptions made in place of such knowledge. This would ultimately slow down the development velocity of the students, as applying such overdue processes would hinder the SDLC progress.

Contents could be better structured to be more applicable to the Team Projects, as both the PSD and Team Projects modules are running concurrently. The synchronisation between these two modules is of utmost importance to attain applied learning effectively. Having a timely correlation between these modules would greatly benefit the absorption and learning of such knowledge.

C. Flexibility of Project Evaluation Rubrics

Currently, there are fixed marking rubrics for Team Projects that are clearly iterated in the project specifications. Given the varying projects described earlier in Section V-(A), the potential improvement directly complements the variance in the software industrial projects. With such a spread in terms of projects and their specifications, having fixed marking rubrics would be unfair to some projects, as the applicability of a certain practice or process might be highly implementable in one project but not the other.

Potential improvements could be made to transition into more flexible marking rubrics or dynamic marking rubrics according to respective projects in the Team Projects. This would make the rubrics more applicable to the allocated software projects and give a better guideline to the students partaking in such projects. Additionally, having more relatable and applicable guidelines could potentially achieve fairer gradings and improved learning.

D. Introduction of In-Demand Practices

In theoretical knowledge learnt in the PSD module, although some widely used industry practices like Agile process and Scrum are taught. But most of the software engineering and software development processes, guidelines, and principles are already out in practice in the industry for quite some time. Having such a safe approach would undoubtedly achieve the fundamental teachings of software engineering, but such teaching methods do not encourage innovation.

With most companies in the software industry conducting on-the-job training for employees and undergraduates, such training indicates the necessity of the latest knowledge [20].
Cutting-edge or in-demand technologies, practices, and methodologies could be evaluated and introduced to better equip undergraduates with industry-relevant skills. For example, upcoming and trending practices like DevOps, automated infrastructure deployment via Infrastructure as Code (IaC), or containerisation practices using Docker and Kubernetes could be introduced, enabling undergraduates to be better equipped for the industry.

VI. CONCLUSION

In the CS joint degree of SIT-UoG, the PSD and Team Projects courses attempt to tackle the problem of unprepared graduates as they enter the workforce. Through our review of existing literature, in general there are three types of current methodologies in teaching the software engineering courses: partnering with industrial experts, project-based learning, and communication-based learning. Their advantages and disadvantages are discussed in detail.

In this paper, the survey studies have been conducted to collect the feedback from three CS cohorts in the University about the PSD and Team Projects modules, with 30 first-year students, 21 penultimate year students, and 21 final year students. The sentiments and opinions of these three cohorts of students on these two courses are analyzed and compared. It is observed from the results that most students participating to the survey feel more prepared at the end of the learning of these two modules in their curriculum. Final year students have overwhelmingly positive sentiments about what is taught in these two modules after going through their internship and industry attachments, which entail real life work experience, despite initially having reservations about how useful these two modules actually are.

However, there is still room for enhancement of the learning of the PSD and Team Projects courses. It is expected to continue iterating and improving these two modules according to the students’ feedback, much like how the Agile manifesto taught to the students. Continuous improvements overtime will lead to a more prepared batch of graduates to software industry at the end of their university study.

REFERENCES