

Title	Evaluation of the effectiveness of developing real-world software projects as a motivational device for bridging theory and practice
Authors	Dong, Dapeng;Butler, Robert;Herbert, John
Publication date	2022-05-03
Original Citation	Dong, D., Butler, R. and Herbert, J. (2022) 'Evaluation of the effectiveness of developing real-world software projects as a motivational device for bridging theory and practice', Journal of Further and Higher Education, 46(9), pp.1275-1289. https://doi.org/10.1080/0309877X.2022.2070727
Type of publication	Article (peer-reviewed)
Link to publisher's version	10.1080/0309877X.2022.2070727
Rights	© 2022, UCU. Published by Routledge - Taylor & Francis Group. This is an Accepted Manuscript version of the following article, accepted for publication in Journal of Further and Higher Education: Dong, D., Butler, R. and Herbert, J. (2022) 'Evaluation of the effectiveness of developing real-world software projects as a motivational device for bridging theory and practice', Journal of Further and Higher Education, doi: 10.1080/0309877X.2022.2070727. It is deposited under the terms of the Creative Commons Attribution-NonCommercial License (http://creativecommons.org/licenses/by-nc/4.0/), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited. - https://creativecommons.org/licenses/by-nc/4.0/
Download date	2026-09-25 10:57:24
Item downloaded from	https://hdl.handle.net/10468/13207



University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Evaluation of the Effectiveness of Developing Real-world Software Projects as a Motivational Device for Bridging Theory and Practice

Dapeng Dong^{a*}, Robert Butler^b and John Herbert^c

^a *Department of Computer Science, Maynooth University, Maynooth, Ireland;*

^b *Department of Economics, University College Cork, Cork, Ireland;*

^c *Department of Computer Science, University College Cork, Cork, Ireland*

*Correspondence Author: Dapeng Dong, e-mail: dapeng.dong@mu.ie

Abstract

While incorporating project-based exercise is a common practice in software engineering education, few studies have been conducted in investigating how real-world project development influences university students' proactive learning and knowledge transformation. This study aims to evaluate the effectiveness of developing real-world projects with industry engagement in encouraging students to apply knowledge to practice and be more proactive in learning. Using a two-group, post-test quasi-experimental design, the performance between the students taking real-world project development and the students in the control group are compared using descriptive statistics, the independent samples *t*-test and Welch *t*-test, accordingly. Both the Spearman's rank-order and Kendall's τ -b are used to examine the relationship between students' practical works and the level of knowledge transformation estimated by the students through online surveys. The results suggest that using real-world projects in the classroom can be an effective motivational device for proactive learning and knowledge transformation. Project-based exercise should be both comprehensive and keeping pace with technology development driven by the software industry evolution to be more effective. The direct interaction with stakeholders, dynamic requirements change, employment of Agile methods, self-organising teams, and using challenging real-world projects, are essential in simulating a real-world software development environment in the classroom.

Keywords: Software engineering; project; development; theory; practice

Introduction

Software engineering is a highly application-oriented subject. Since the term software engineering was first coined in 1968 (Naur, 1968), the field has evolved rapidly covering virtually all aspects of software development, such as requirements engineering, architectural design, software testing, configuration management and project management. This diversity often led lecturers to be selective when deciding on course content. It inevitably creates knowledge gaps when teaching software engineering.

With the ever-increasing need for efficient and effective methods for the development of versatile software applications, methodologies, and associated techniques and tools are continuously developed. For example, from the early plan-driven Waterfall methodology (Royce, 1987) to the recent incremental process and Agile methods ¹. This evolution has made software engineering topics fluid and dynamic, which has added another dimension to the effective teaching of software engineering courses. In addition, software development methods taught in the classroom are often abstract models, generalised guidelines, best practices and frameworks (Sommerville, 2011), this abstraction is the third dimension that challenges many lecturers and creates gaps for students to perceive the connection between abstract models and their applications in real-world software project development.

¹ <http://agilemanifesto.org>

It is widely acknowledged that doing practical exercise remains one of the most effective approaches in teaching software engineering courses, thus the use of mixed lecturing and laboratory work with an emphasis on project development has been the predominant mode of instruction in many educational settings (Mas, 2020) (Gary, 2009) (Klappholz, 2009) (Demurjian, 2009) (Robillard, 1988). In our undergraduate courses, we have implemented the same mode of instruction. In particular, we strive to design student's projects that ensemble as many characteristics of real-world software development as possible, for example, role-playing and Scrum teams (Schwaber, 1997). However, due to various constraints, such as the length of the semester and the credits allocated to written work and practical work, the project scenarios are often greatly simplified. More crucially, the interaction with real stakeholders, which is one of the most important aspects in software engineering, is missing. As a side-effect, the students do not seem to be motivated. In response to this, we have established partnerships with local small and medium enterprises (SMEs) ¹, with our intentions to bring real-world project development to our students and to simulate a real-world software project development environment in the classroom.

Simulating real-world software development environment

It is generally accepted that software engineering processes and principles are most needed *"when the size and complexity of the software project are so great that the project surpasses the comprehension of any single person"* (Henry, 1983). In collaboration with industry partners, we strive to provide a simulated real-world

¹ The establishment of the partnership was initially bridged through the Experiential Learning Office of the University.

software development environment, which consists of five important components, including (1) direct interaction with stakeholders, (2) dynamic requirements change, (3) application of Agile development methods, (4) self-organising teams, and (5) relatively challenging real-world project that could only be achieved in teams with proper planning.

During integration of the project, the structure of the module is slightly modified. The main components of the module are lecturing and practicals. Theories are delivered mainly through lectures and discussion sessions, and are further enhanced and extended through reading assignments on a per topic basis. The main theme of the practice that was previously focused on exercising discrete software engineering topics, are now changed to fully integrated project development of iterative processes. More specifically, in accordance with the length of the semester, the actual development of the project is divided into 5 *Sprints*¹, i.e., 2 weeks per Sprint. Students are self-grouped into small and self-organising teams (4 - 6 members per team). In each Sprint, students need to set a Sprint goal and to complete a set of requirements prioritised by the team members, following the development cycle of analysis, design, code and feedback. The conventional weekly laboratory sessions are reserved for team members to meet each other, to create Sprint plans, to ask questions, to seek help from instructors and the lecturer, and to review the previous Sprint. Teams manage their own time to achieve the Sprint goal within the 2 weeks.

¹ In Scrum software development method, a *Sprint* is a short duration of the project development that contains activities including planning, development, review and retrospective.

In addition, students are required to write Project Development Journals that document what went well, what went wrong, and what could be improved in the next Sprint, on a per Sprint basis, in a collaborative manner. In this study, two projects are made available to the students. One project proposal was jointly written by the industry partner and the lecturer. This real project has been carefully scoped and made to be self-contained, i.e., it can be completed without other services to function properly. The other project scenario was fabricated by the lecturer. The written works and the implementation of the project reflect the academic performance of the students. The students have complete freedom to choose one of the projects for the module.

It is advised that students need to be well-prepared prior to the project development (Cuthbert, 1995). At the beginning of the semester, senior staff of the industry partner are invited to the classroom to introduce their business context, to clarify the scope of the project, and to facilitate the requirements gathering process. At certain stages of the development, students also need to present their work to the stakeholders, and subsequently improve their work based on the feedback received. Direct stakeholders from the industry partner are also invited to join live discussion sessions with the students to clarify unclear requirements and any ambiguities encountered. Lectures are carefully planned and aligned with the project development stages. The activities and their organisations simulate a real-world software development environment. The detailed structure of the module is shown in Figure 1. The detailed timeline and the progress of activities are listed in Appendix A.

Insert Figure 1 here

Research questions

We believe that introducing real-world project development to the classroom can be an effective way for students to think/rethink and to frame/reframe their epistemic knowledge and more importantly to apply their epistemic knowledge to practice. In this initial stage of our study, we aim to answer the following research questions.

- (1) **RQ1:** Will developing real-world software projects contribute to students' interest in active learning?
- (2) **RQ2:** Will developing real-world software projects help students acquire extra knowledge/skills and build a more complete conceptual network of the subject domain?
- (3) **RQ3:** Will students be more inclined to apply knowledge learned in class to practice during the process of developing real-world software projects?

Literature Review

Teaching software engineering is a challenge. In the extensive literature (Mas, 2020) (Kuhrmann, 2016) (Cagiltay, 2007) (Ohlsson, 1995) (Robillard, 1988) (Henry, 1983), two main reasons emerge. First, the academic content is often a summary of field experiences and taught at an abstract level. The introduction to and explanation of the fundamental knowledge are largely conditional on situations, e.g., there is no *golden rule* for the application of software process models to real project development rather it *depends on the situation*. It is hard for students to see the direct connection or the usefulness of the academic content in real-world applications, thus, students are often

left in a situation where, when given a project scenario, they do not know what to do or do it in an ad-hoc manner. Second, technologies are rapidly developing and evolving with industry trends while expanding to include much broader areas of study that makes Software Engineering an interdisciplinary subject, consequently, leads it to be a *less focused* subject. The vast amount of information is overwhelming. The topics are more discrete and less coherent. As a result, students are often not motivated or show low interest in the subject. On the surface, the characteristics of the abstraction, the diversity and the dynamism make teaching software engineering hard, but fundamentally, it is the connection between theory and practice that challenges the overall effective teaching and learning.

In response, many studies have been conducted and various approaches have been reported, but all merge to some variant of project-based teaching and learning practice with each emphasizing different aspects. The authors in (Kuhrmann, 2016) argue that students could be trained to deal with stressful and exceptional situations more effectively when working in a self-organising and cross-functional team. In (Williams, 2002) the authors report that collaborative learning in pairs may enhance students' higher-order thinking skills than students who worked alone. In an effort to increase students' interest in software engineering, many educators have tried to embed game-inspired exercises in their teaching and the results reported are promising (Cagiltay, 2007) (Sindre, 2003). In addition, in our view, depending on whether a module covers more generic topics (e.g., introduction to software engineering, software process models, architectural design or project management) or more focused topics (e.g.,

programming languages, design patterns or software testing), the type of project and the technologies to be exercised should be thoughtfully considered.

In the former case, using trending technologies may be better for preparing our students to be job ready; in the latter case, technologies should be specific to the topic being taught. These considerations are especially true with the recent development of new software development methodologies, and the use of Agile methods has been predominant in, but not limited to, the software industry sector. Agile development using Scrum (Mahnic, 2011), eXtreme Programming (XP) (Anslow, 2015; Anslow, 2015), Lean (Chatley, 2017), hybrid methods (Kropp, 2014) are reported successful in software engineering education.

In reviewed literature in project-based teaching and learning, at a high level, a common pattern becomes apparent that is the incorporation of teamwork and Agile methods, but fewer have realised the importance of industry engagement. In an experiment of using the Lean software development method, the authors (Chatley, 2017) point out that experienced industry partners may be better suited for teaching practical skills, whereas academic content should be delivered by lecturers, so that the collaboration could *"improve the perceived relevance of the course material and adapt curriculum to align with industry needs"*. In fact, the need for industry engagement has long been recognised since the early 1980's by (Henry, 1983) (Robillard, 1988). Furthermore, the authors also emphasize the necessity of the complexity of student projects, i.e., practical work must present a certain level of complexity to be effective, however, this has been a largely neglected issue (Ralph, 2018).

In addition, we think engaging industry practitioners should not be limited to imparting of skills, but also the interaction with project stakeholders of broader interest (e.g., business management teams, product owners and project managers) in which we believe is one of the most important aspects that facilitates students to develop and improve essential transferable skills, such as, negotiation, meeting, communication and presentation.

Undoubtedly, practices and methods reported in literature are successful and effective, but additionally, from many years of working and collaborating with industry partners, we think that another vitally important component is still missing, that is the dynamic requirements change. It is the dynamic requirements change that makes the success of a project development a variable. This is also one of the core issues that many modern Agile methods are trying to cope with. Thus, in simulating a real-world software development environment, apart from the common patterns used by many educators, the direct interaction with stakeholders, the dynamic requirements change, and the complexity of the project are essential. In our implementation of teaching strategy for the Software Engineering and Software Process module, we include all these essential components.

Method

Participants and context

Data for this study are obtained from students taking the Software Engineering and Software Process module offered by the Computer Science Department of the

National University of Ireland Maynooth. In total, 149 students from two second-year BSc. majors, Computer Science Software Engineering (CSSE) and Multimedia Web Development (MMWD), enrolled in the module took part in the study. Of the 147 students, 61 students from CSSE and 86 students from MMWD participated in the study. The age of the students ranges from 18 to 21.

Research design

A two-group, post-test quasi-experimental design is used in the study. At the beginning of the semester, two projects were made available to the students. One project idea is proposed by the industry partner [company name blinded] and another project scenario is fabricated. Both projects are self-contained. The scope and the assessment criteria of the projects are well-defined and balanced. The two projects are different in two major aspects:

- (1) The students taking the real-world software development project (coded as *competition* project) have opportunities to interact with external stakeholders throughout the development process. Unclear requirements and ambiguities are directly clarified by external stakeholders, whereas issues associated with the fabricated project (coded as *non-competition* project) are answered by the lecturer.
- (2) The progress of the students taking the non-competition project is monitored by the lecturer. In comparison, the progress of the development of the competition project is reported to the external stakeholders regularly, through presentation, project discussion sessions and a project management platform. Based on the

feedback received, the students need to adjust and improve their work accordingly. More importantly, the requirements of the competition project are (slightly) evolving as the external stakeholder gradually realised what is truly needed. This requirements change is an important component in the simulation of a real-world development environment in the classroom, but it indeed makes the project challenging.

The students are given a choice to select one of the projects based on their own interest, as part of the module. A two-week grace period is also given to the students for them to discover whether their choice fits into their timetable (e.g., some meetings are held during the weekend) and expectations (e.g., whether they find the project interesting or too difficult to achieve, especially for the competition project). Initially, 59.2% of the students selected the competition project and 40.8% the non-competition project. After the grace period, the figure has changed to and stabilised at 41.2% (competition project) and 58.8% (non-competition project). The reasons why students chose different projects are collected through an anonymous online survey, as shown in Table 1. 116 students participated in the survey.

With this specific quasi-experimental design, the validity of the results of the subsequent analysis relies heavily on the assumption that the two groups of students are homogeneous (McMillan, 2010). We are particularly concerned with the group of students who considered the competition project difficult. In response to this, a test with a mix of 18 questions that are suitably tailored to assess factual knowledge,

conceptual understanding and applications, is given to the students prior to the project development. The group descriptive statistics of the test results are shown in Table 2.

To determine the existence of differences between the two groups of students, the Independent Samples *t*-test is performed. According to (Pallant, 2010) (Gravetter, 2020) (Ghasemi, 2012), with the sample size given in Table 2, the test for normality is not mandatory. However, it can be seen that the sample sizes of the two groups are unequal ($N_{competition} = 61$ and $N_{non-competition} = 86$), thus we performed the Levene's test of homogeneity of variances. The results ($F(0.53)$, $p = 0.47$) indicate that the variances for the two groups are equal, thereby fulfilling the assumption for the Independent Samples *t*-test. In accordance with the results shown in Table 2, no significant difference is observed ($t = -0.424$, $p = 0.672$). Therefore, the group of the students taking the competition project is considered as in the experimental group and the group of the students taking the non-competition project is considered as in the control group.

Insert Table 1 here

Data analysis

In the study, we take a quantitative approach to data analysis using both descriptive and inferential statistical methods. Since the homogeneity of variances can significantly reduce the power of many statistical methods, we first conduct a homogeneity of variances test using the Levene's test (Gleser, 2012). If the *null* hypothesis of equal variances is rejected, the Welch *t*-test (Welch, 1947) will be used

to test statistical difference between the means of the data of the two groups, otherwise, the Independent Samples *t*-test will be used. Note that according to (Pallant, 2010) (Gravetter, 2020) (Ghasemi, 2012), the normality of the data is not our main concern with the sample size given in the experiments. Correlation analysis is performed to examine how the students' performance in practical work is associated with the level of knowledge applied to practice.

In the analysis, the acceptance level of significance is set to $p < 0.05$.

Results

Improving student active learning (RQ1)

To determine whether developing a real-world project contributes to students' interest in active learning, a short online survey was conducted regarding the average coding hours, reading hours, and project meeting frequency per week, at the end of the semester. It was an anonymous online survey. 116 students participated in the survey. The results are shown in Figure 2. To determine whether there is a significant difference in active learning between the two groups, we first performed the Levene's test on the data. The Levene's tests for *Coding Hours* and *Reading Hours* are significant ($p < 0.5$), suggesting a violation of the equal variance assumption, as shown in Table 3, thereby, the Welch *t*-test was used.

The alternative hypothesis of the Welch t -test specifies that the students taking the competition project spent more time on coding the project ($mean = 6.24, sd = 4.95$) and more time on reading course related materials ($mean = 7.5, sd = 5.72$), compared with the students taking the non-competition project for *Coding Hours* ($mean = 4.57, sd = 2.34$) and *Reading Hours* ($mean = 4.31, sd = 2.99$). This is also supported by student feedback: "*I found it was interesting. The competition has driven us to learn more and more.*" In addition, team members in the control group tend to meet as frequently as the teams in the experimental group ($p = 0.762$) excluding meetings with external stakeholders.

Insert Figure 2 here

Insert Table 3 here

Helping students extend knowledge network (RQ2)

During the project development, apart from the help received from the lecturer and instructors, the students are also suggested to search for extra information, related materials or tutorials from various sources, to extend their knowledge network. We hypothesize that developing real-world software projects helps students acquire extra knowledge/skills and build a more complete conceptual network of the subject domain, compared with the students in the control group. In the same survey as discussed in the previous section, the students were asked to give direct answers to the question, and the results are shown in Figure 3. The result from the χ^2 -square test ($\chi^2 = 8.796, df = 1, p < 0.005$) suggests that our hypothesis is supported.

Insert Figure 3 here

Insert Table 4 here

Furthermore, a short test with 7 open-ended, analysis-oriented questions was given to the students after the midterm break of the semester. Table 4 shows the group descriptive statistics on the test results and the results from the Welch t -test. The t -test results in a t value of -2.135, which is significant ($p < 0.05$). Since the effect size is close to the suggested medium level (0.5), thus our hypothesis is supported by the t -test, i.e., working with real-world projects may help (or *force*) students to learn extra knowledge/skills and to build a more complete conceptual network of the subject domain, consequently, improve their understanding of the subject matter and argument analysis skills.

Encouraging students to apply knowledge to practice (RQ3)

One of our main intentions for bringing a real-world software development project to the classroom is to encourage students to apply knowledge/skills learned in class to practice. In a separate survey conducted at the end of the project development, students were asked to rate how much knowledge/skills learned in class is applied to the project development. 99 students participated in the survey, and were asked to rate from 1 (almost nothing) to 10 (almost everything). The results are shown in Table 5 and the

rating distributions are shown in Figure 4. The mean ranks, as shown in Table 5, are calculated for the Mann Whitney test.

Insert Figure 4 here

Insert Table 5 here

Insert Table 6 here

According to (De Winter, 2010), the Independent Samples *t*-test and Mann Whitney test are both effective for the analysis of Likert scale data. However, we are concerned with the distribution of the data from the experimental group, as shown in Figure 4, i.e., the rating results present a bimodal distribution. A potential cause of this may be the self-organising teams. As in the study, students were asked to form teams by themselves, so that team members are familiar with each other, thus we hope that the teaming would be more effective. One possible side effect could be that some team members are close friends having different levels of motivation and interest in the project, and some students rely on their team members, hence the bimodal distribution. To this end, both the Independent Samples *t*-test and the Mann Whitney test are performed, and the results are shown in Table 6. As can be seen, there is a significant difference ($p < 0.05$ from both tests) in that students taking the real-world project development tend to apply theory to practice more actively than the students taking the fabricated project. Furthermore, the difference between the two-group means is greater than 0.2 standard deviations, which is non-negligible (Cohen, 2013), indicating a small effect from the Mann Whitney test and a medium effect from the Independent Samples

t-test, respectively. The cause of the comparison result may be in the complexity of the real-world project that *forces* the students to use suitable methodologies, or the students are proactively trying to apply the knowledge to practice as in the simulated real-world development environment.

Discussion and Implications

In this study, a real-world project is introduced to the classroom with the main intention to encourage students to apply theory to practice. Previously, we also had project work as part of the module, but those were fabricated and greatly simplified. The fabrication and simplicity led our students to feel that there is no need for the application of sophisticated and abstract methods, techniques, best practices, guidelines and/or frameworks in the development. This is also in line with the experience shared by (Ralph, 2018). Furthermore, as projects were fabricated, the project requirements were crystal clear and static, which is quite contrary to real-world situations. We also observed that the majority of the students develop software project impromptu. In this regard, we believe that a student project and its organisation need to be comprehensive, i.e., it should include the five essential components identified above, to be more effective. In addition, considering the external factors such as the rapid development of technologies in the information & communications technology (ICT) sector and the dynamic job market, employers are really looking for students having up-to-date knowledge, ability to use new technologies and transferable skills. The students, our future IT professionals, need to see and to experience the complexity and the challenges that exist in real-world application development.

From the students' feedback, we also see that the industry engagement is an effective motivational device that helps students transform epistemic knowledge to craft knowledge in studying software engineering, for example:

- *"I'm quite into it, ... I really want to know how it runs and what sort of questions the industry people ask."*
- *"Since completing your module, I managed to acquire a job as a Software Engineer in a company called [company name blinded]. In the interview, I was asked about any projects that I had completed, and I immediately started talking about the industry project we had been working on with [company name blinded]. I am absolutely certain that that was a massive aid for in the interview..."*

Our findings show that the students working on real-world projects are more proactive in learning, i.e., the students in the experimental group spent significantly more time on coding the project and reading course related materials. This kind of learning by doing approach (Winn, 1995) can help students better understand the subject topics. This is supported by the results from a short test with open-ended, analysis-oriented questions, as outlined above. The conclusion is consistent with (Klappholz, 2009). The motivation and the proactive learning resulting from industry engagement served as a foundation for engaged learning in the class (Biggs, 2011).

Insert Table 7 here

From analysis of the survey data, we confirm that students in the experimental group are more inclined to apply knowledge learned to practice. Furthermore, to determine the relationship between how much the students apply knowledge to practice and the aggregated continuous assessment results including the results from the implementation of the system, the project design document, and the project development journal (statistics on the results are shown in Table 7), i.e., how strongly the two factors are monotonously related, both the Spearman's rank-order and Kendall's τ -b correlation are performed. Note that since the Likert scale can be considered as ordinal data, thus, according to (Khamis, 2008), either the Spearman's rank-order and Kendall's τ -b correlation is effective.

The results from both the Spearman's rank-order and the Kendall's τ -b indicate a positive correlation between the two factors in the experimental group, which is statistically significant ($\rho = 0.434$, $p = 0.008$ and τ -b = 0.032, $p = 0.009$, respectively), as shown in Figure 5a. This is understandable as students are more active in their coding and researching process. In contrast, Figure 5b shows that the level of applying theory to practice is slightly negatively correlated with the continuous assessment results, but statistically insignificant ($\rho = -0.211$, $p = 0.096$ and τ -b = -0.156, $p = 0.093$, respectively). This may be explained by the fact that the non-competition project is relatively generic, and its requirements are clearly defined, thus the students might feel there is little need for the application of theory to practice. Although the difference between the continuous assessment results of the experimental group and the control group is statistically significant, obtained from the Mann Whitney test ($t = 684.0$, $p =$

0.0005), the average score of the control group is still up to par with the students in the experimental group, as shown in Table 7. This may suggest we need to reconsider the evaluation criteria for the module in general. For example, in this particular module, the majority of the continuous assessments were evaluated based on the quality and the completeness of the project and its associated documentation. To what extent the students applied their epistemic knowledge to the project development was not directly considered or measured. In fact, it is another open question how this could be measured.

In addition, the development of the project in teams serves as a virtual integration that integrates knowledge, technical skills, and transferable skills, such as, communication, team working, management and creative problem-solving. It should not be

Insert Figure 5a here

Insert Figure 5b here

overlooked that, in some cases, students have difficulties when applying knowledge to practice. In teaching this module, instructional scaffolding is provided by means of assistance from instructors, imparting field experience from invited guest lecturers, and a reference implementation of a skeleton of the project. An in-depth study in instructional scaffolding in Software Engineering courses will be reported in the future work. But, perhaps, the most concerning aspect of using real-world projects is its suitability, i.e., how to sustain engaged learning in the absence of collaboration with

external partners. This certainly requires a continued support from the department or the university where the initial connection with industry partners can be facilitated and the associated legal issues (such as, the ownership of inventions, copyright of artistic works and source code) can be addressed, thus lecturers can focus on teaching and the integration of the projects.

Conclusion and Limitations

This study focuses on evaluation of the effectiveness of using real-world project development in the classroom for bridging the gap between theory and practice. Although, project-based exercise has been predominant in software engineering education, the uniqueness of this work lies in the use of the combination of the five essential components identified in the paper, i.e., the direct interaction with stakeholders, dynamic requirements change, complexity of the project, application of Agile methods and self-organising teams, in simulating a real-world software project development environment in the classroom, in an effort to encourage students to apply epistemic knowledge to practice. This research example illustrates the practicality and the effectiveness of the proposed method for teaching software engineering courses, especially for those topics taught at an abstract level. Using the two-group, post-test, quasi-experimental design, we have compared the students' academic performance between the experimental and control groups. Our findings show that industry engagement is an effective motivational device for stimulating student proactive learning and extending their knowledge network. Furthermore, students are more inclined to apply epistemic knowledge to practice when developing real-world projects. Although the overall result is promising, we must be aware of a potential issue

associated with the types of projects used in the classroom. In a continued collaboration with industry partners, we will likely receive different projects every year. Generally, each project has its own set of characteristics, even for a development of the same type of software applications. This may vary the effectiveness of the teaching strategy as outlined in the paper. Thus, it is vitally important for the lecturers to carefully select projects that are both interesting and attainable. A longitudinal study has been planned. The effectiveness of the practice and potential adjustment to the teaching strategy will continuously be reported to the community.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This research was supported by the Project Live Initiative, which was funded by HEA

Bibliography

Anslow, C. & M. F., 2015. *An experience report at teaching a group based agile software development project course.* Kansas, Missouri, ACM, pp. 500-505.

Biggs, J. & T. C., 2011. *Teaching for quality learning at university.* UK: McGraw-hill education.

Cagiltay, N. E., 2007. Teaching software engineering by means of computer-game development: Challenges and opportunities. *British Journal of Educational Technology*, pp. 38(3), 405-415.

Chatley, R. & F. T., 2017. *Lean learning-applying lean techniques to improve software engineering education.* Buenos Aires, IEEE, pp. 117-126.

Cohen, J., 2013. *Statistical power analysis for the behavioral sciences.* New York: Routledge.

Cuthbert, K., 1995. Project planning and the promotion of self-regulated learning: From theory to practice. *Studies in higher education*, pp. 20(3), 267-277.

De Winter, J. F. C. & D. D., 2010. Five-point likert items: t test versus Mann-Whitney-Wilcoxon. *Practical Assessment, Research, and Evaluation*, pp. 15(1), 11.

Demurjian, S. A. & N. D. M., 2009. Experiences in Project-Based Software Engineering: What Works, What Doesn't. In: *Software Engineering: Effective Teaching and Learning Approaches and Practices.* Hershey, PA: IGI Global, pp. 191-211.

- Gary, K. A., 2009. The Software Enterprise: preparing industry-ready software engineers. In: *Software Engineering: Effective Teaching and Learning Approaches and Practices*. Hershey, PA: IGI Global, pp. 115-135.
- Ghasemi, A. & Z. S., 2012. Normality tests for statistical analysis: a guide for non-statisticians. *International journal of endocrinology and metabolism*, pp. 10(2), 486.
- Gleser, L. J. P. M. D. P. S. J. & S. A. R., 2012. *Contributions to probability and statistics*. New York: Springer Science & Business Media.
- Gravetter, F. J. W. L. B. F. L. A. B. & W. J. E., 2020. *Essentials of statistics for the behavioral sciences*. USA: Cengage Learning.
- Henry, S., 1983. A project oriented course on software engineering. *ACM SIGCSE Bulletin*, pp. 15(1), 57-61.
- Khamis, H., 2008. Measures of association: how to choose?. *Journal of Diagnostic Medical Sonography*, pp. 24(3), 155-162.
- Klappholz, D. A. V. L. M. K. O. C. & J. A., 2009. A framework for success in real projects for real clients courses. In: *Software Engineering: Effective Teaching and Learning Approaches and Practices*. Hershey, PA: IGI Gloal, pp. 157-190.
- Kropp, M. & M. A., 2014. *New sustainable teaching approaches in software engineering education*. Istanbul, Turkey, IEEE, pp. 1019-1022.
- Kuhrmann, M. & M. J., 2016. *When teams go crazy: An environment to experience group dynamics in software project management courses*. Austin, TX, IEEE, pp. 412-421.
- Mahnic, V., 2011. A capstone course on agile software development using scrum. *IEEE Transactions on Education*, pp. 55(1), 99-106.
- Mas, A. M. A. L. & C.-P. R., 2020. Enhancing the Student Perception on Software Project Management in Computer Science. *IEEE Transactions on Education*, pp. 64(1), 1-11.
- McMillan, J. H. & S. S., 2010. *Research in Education: Evidence-Based Inquiry, MyEducationLab Series*. USA: Pearson.
- Naur, P. & R. B., 1968. *Software engineering: Report of a conference sponsored by the nato science committee*, Garmisch, Germany: Scientific Affairs Division, NATO.
- Ohlsson, L. & J. C., 1995. A practice driven approach to software engineering education. *IEEE Transactions on Education*, pp. 38(3), 291-295.
- Pallant, J., 2010. SPSS survival manual: a step by step guide to data analysis using SPSS. In: London: Routledge, p. p.198.
- Ralph, P., 2018. *Re-imagining a course in software project management*. Gothenburg, Sweden, In Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training, ACM, pp. 116-125.
- Robillard, P. N. & L. D., 1988. The simulated working environment in a project-based software engineering course. *Computers & Education*, pp. 12(4), 471-477.
- Royce, W. W., 1987. *Managing the development of large software systems: concepts and techniques*. Monterey, CA, In Proceedings of the 9th International Conference on Software Engineering, ACM, pp. 328-338.
- Schwaber, K., 1997. Scrum development process. In: *Business object design and implementation*. London: Springer, pp. 117-134.
- Sindre, G. L. S. & V. O. V., 2003. *Positive experiences with an open project assignment in an introductory programming course*. Portland, OR, IEEE, pp. 608-613.
- Sommerville, I., 2011. *Software engineering 9th Edition*. UK: Pearson.
- Welch, B. L., 1947. The generalization of 'STUDENT'S' problem when several different population variances are involved. *Biometrika*, pp. 34(1-2), 28-35.

Williams, L. W. E. Y. K. F. M. & M. C., 2002. In support of pair programming in the introductory computer science course. *Computer Science Education*, pp. 12(3), 197-212.

Winn, S., 1995. Learning by doing: Teaching research methods through student participation in a commissioned research project. *Studies in Higher Education*, pp. 20(2), 203-214.

Experiential Learning Office.

References

1. Bibliography

Anslow, C. & M. F., 2015. *An experience report at teaching a group based agile software development project course.*. Kansas, Missouri, ACM, pp. 500-505.

Biggs, J. & T. C., 2011. *Teaching for quality learning at university*. UK: McGraw-hill education.

Cagiltay, N. E., 2007. Teaching software engineering by means of computer-game development: Challenges and opportunities. *British Journal of Educational Technology*, pp. 38(3), 405-415.

Chatley, R. & F. T., 2017. *Lean learning-applying lean techniques to improve software engineering education*. Buenos Aires, IEEE, pp. 117-126.

Cohen, J., 2013. *Statistical power analysis for the behavioral sciences*. New York: Routledge.

Cuthbert, K., 1995. Project planning and the promotion of self-regulated learning: From theory to practice. *Studies in higher education*, pp. 20(3), 267-277.

De Winter, J. F. C. & D. D., 2010. Five-point likert items: t test versus Mann-Whitney-Wilcoxon. *Practical Assessment, Research, and Evaluation*, pp. 15(1), 11.

Demurjian, S. A. & N. D. M., 2009. Experiences in Project-Based Software Engineering: What Works, What Doesn't. In: *Software Engineering: Effective Teaching and Learning Approaches and Practices*. Hershey, PA: IGI Global, pp. 191-211.

Gary, K. A., 2009. The Software Enterprise: preparing industry-ready software engineers. In: *Software Engineering: Effective Teaching and Learning Approaches and Practices*. Hershey, PA: IGI Global, pp. 115-135.

Ghasemi, A. & Z. S., 2012. Normality tests for statistical analysis: a guide for non-statisticians. *International journal of endocrinology and metabolism*, pp. 10(2), 486.

Gleser, L. J. P. M. D. P. S. J. & S. A. R., 2012. *Contributions to probability and statistics*. New York: Springer Science & Business Media.

Gravetter, F. J. W. L. B. F. L. A. B. & W. J. E., 2020. *Essentials of statistics for the behavioral sciences*. USA: Cengage Learning.

Henry, S., 1983. A project oriented course on software engineering. *ACM SIGCSE Bulletin*, pp. 15(1), 57-61.

Khamis, H., 2008. Measures of association: how to choose?. *Journal of Diagnostic Medical Sonography*, pp. 24(3), 155-162.

Klappholz, D. A. V. L. M. K. O. C. & J. A., 2009. A framework for success in real projects for real clients courses. In: *Software Engineering: Effective Teaching and Learning Approaches and Practices*. Hershey, PA: IGI Gloal, pp. 157-190.

- Kropp, M. & M. A., 2014. *New sustainable teaching approaches in software engineering education*. Istanbul, Turkey, IEEE, pp. 1019-1022.
- Kuhrmann, M. & M. J., 2016. *When teams go crazy: An environment to experience group dynamics in software project management courses*. Austin, TX, IEEE, pp. 412-421.
- Mahnic, V., 2011. A capstone course on agile software development using scrum. *IEEE Transactions on Education*, pp. 55(1), 99-106.
- Mas, A. M. A. L. & C.-P. R., 2020. Enhancing the Student Perception on Software Project Management in Computer Science. *IEEE Transactions on Education*, pp. 64(1), 1-11.
- McMillan, J. H. & S. S., 2010. *Research in Education: Evidence-Based Inquiry, MyEducationLab Series*. USA: Pearson.
- Naur, P. & R. B., 1968. *Software engineering: Report of a conference sponsored by the nato science committee*, Garmisch, Germany: Scientific Affairs Division, NATO.
- Ohlsson, L. & J. C., 1995. A practice driven approach to software engineering education. *IEEE Transactions on Education*, pp. 38(3), 291-295.
- Pallant, J., 2010. SPSS survival manual: a step by step guide to data analysis using SPSS. In: London: Routledge, p. 198.
- Ralph, P., 2018. *Re-imagining a course in software project management*. Gothenburg, Sweden, In Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training, ACM, pp. 116-125.
- Robillard, P. N. & L. D., 1988. The simulated working environment in a project-based software engineering course. *Computers & Education*, pp. 12(4), 471-477.
- Royce, W. W., 1987. *Managing the development of large software systems: concepts and techniques*. Monterey, CA, In Proceedings of the 9th International Conference on Software Engineering, ACM, pp. 328-338.
- Schwaber, K., 1997. Scrum development process. In: *Business object design and implementation*. London: Springer, pp. 117-134.
- Sindre, G. L. S. & V. O. V., 2003. *Positive experiences with an open project assignment in an introductory programming course*. Portland, OR, IEEE, pp. 608-613.
- Sommerville, I., 2011. *Software engineering 9th Edition*. UK: Pearson.
- Welch, B. L., 1947. The generalization of 'STUDENT'S' problem when several different population variances are involved. *Biometrika*, pp. 34(1-2), 28-35.
- Williams, L. W. E. Y. K. F. M. & M. C., 2002. In support of pair programming in the introductory computer science course. *Computer Science Education*, pp. 12(3), 197-212.
- Winn, S., 1995. Learning by doing: Teaching research methods through student participation in a commissioned research project. *Studies in Higher Education*, pp. 20(2), 203-214.

Appendices

Appendix A. Brief Timeline and Progression of Activities

The semester is 16 weeks long. The module consists of two components: lectures (50%) and practicals (50%). The practical work is further divided into four components: project implementation (20%), project documentation (15%), project development journal (10%), and presentation (5%). The project development is aligned with the lectures as outlined below.

- Week 1 - 2 (Project Preparation): (1) External partners are invited to the class to introduce their business context, the vision, the scope, and the expectations of the project. (2) Students are grouped into self-organising teams and assigned to roles. (3) A feasibility report (technical aspect) is generated on a per team basis. (4) Resources are secured by the lecturer and the external partner; the development environment is built and configured by the students.
- Week 3 (Requirements Engineering): (1) Based on the project description and the record of the initial meeting with the external partner, students are asked to outline a list of functional and non-functional requirements for the project (on a per team basis). (2) All teams will share their requirements in the class and then a single list of requirements is compiled through an internal discussion session. (3) External partner is invited to join a workshop/discussion forum to validate and improve the requirements. (4) The first version of the requirements is finalised and stored in a shared, centralised *Product Backlog*.
- Week 4 (Analysis and Design): (1) Each team is asked to provide their own design in terms software system architectures. (2) Design ideas are discussed in

an internal discussion session. (3) Top 10 designs are communicated with the external partner through a workshop at the end of the week, and top 5 designs are kept. (4) All teams need to choose one of the five top designs for the subsequent implementation.

- Week 5 - 14 (Implementation and Testing): The implementation is divided into 5 Scrum Sprints (a biweekly development cycle). (1) In each Sprint, each team needs to set a Sprint goal and specifies how many Sprint points (number of requirements/tasks having different level of difficulties) to burn. (2) At the end of each Sprint, the external partner is invited to join a discussion session and to provide feedback to the students. (3) The process iterates until all requirements are completed.
- Week 15 - 16 (System Testing and Deployment): At the end of the development, all components are integrated, and a system-level test is then performed. Each team presents their final work to the external partner. Feedback from both the lecturer and the external partner is given to the students. One top team is selected by the external partner and the team's work is extended over the summer break (as a development-based micro-internship). Financial support to the team is provided through the allocated funding.

The organisation of the processes and activities simulates a real-world software development environment.

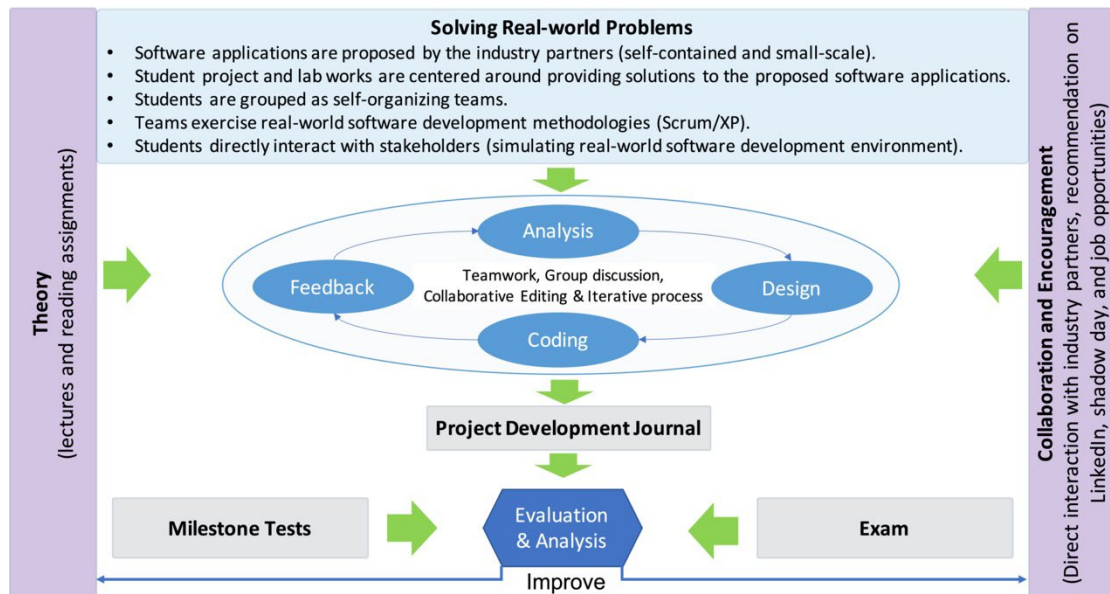


Figure 1. The teaching strategy and the module structure.

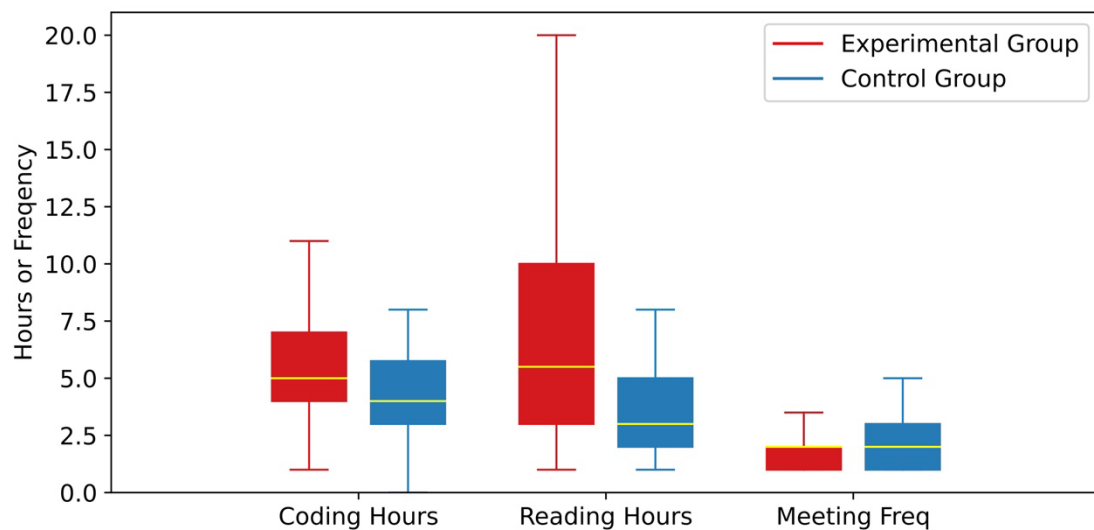


Figure 2. Comparison of student active learning. 116 students participated in the anonymous online survey.

Whether working with the project has helped the students to learn extra knowledge/skills and to build a more complete conceptual network of the subject domain.

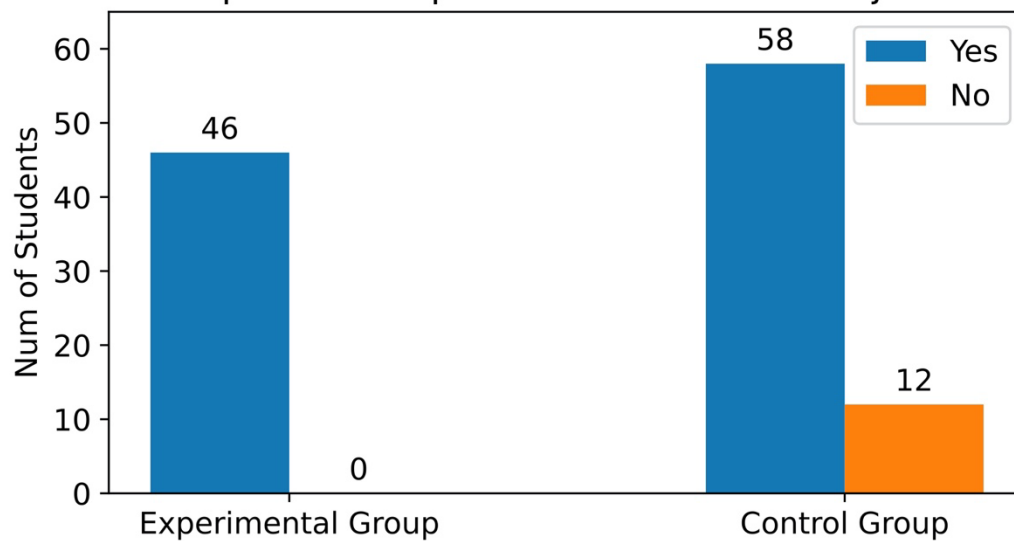


Figure 3. Comparison of the two groups regarding whether the students have learned extra knowledge/skills and have established a more complete conceptual network of the subject domain, through the process of project development. 116 students participated in this anonymous survey.

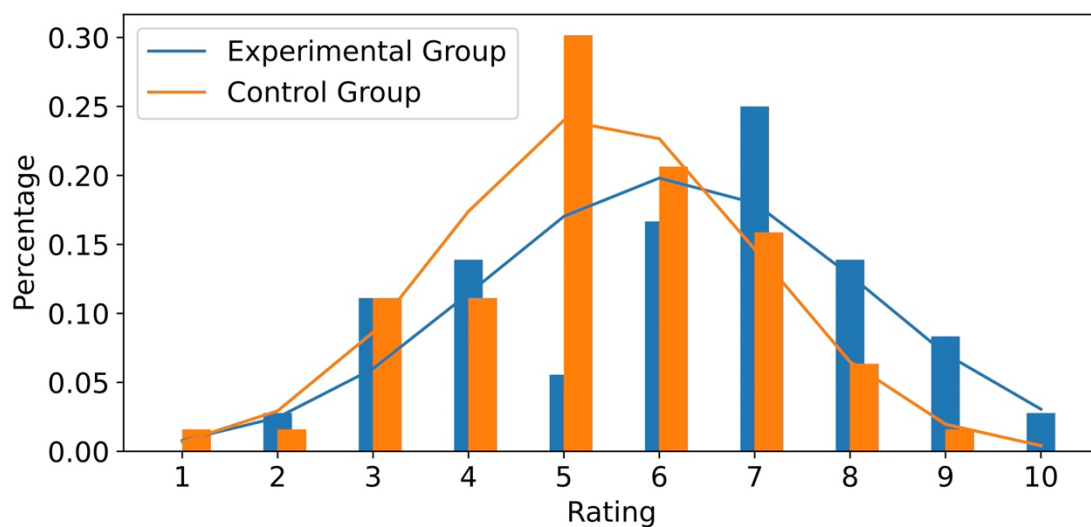
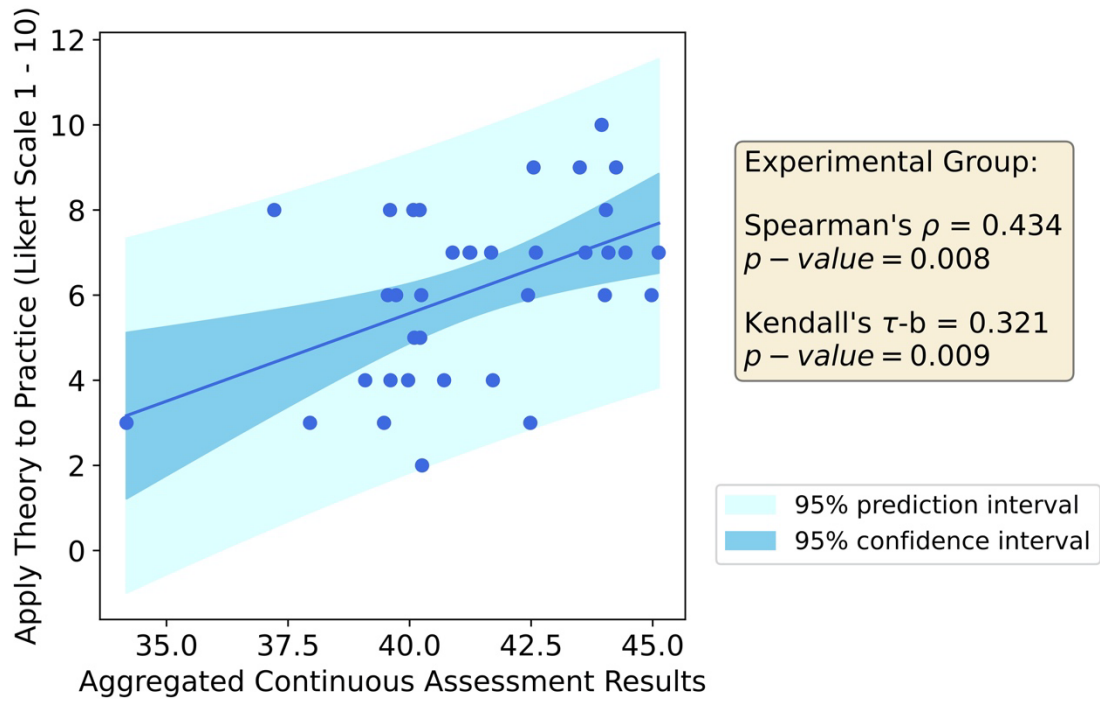
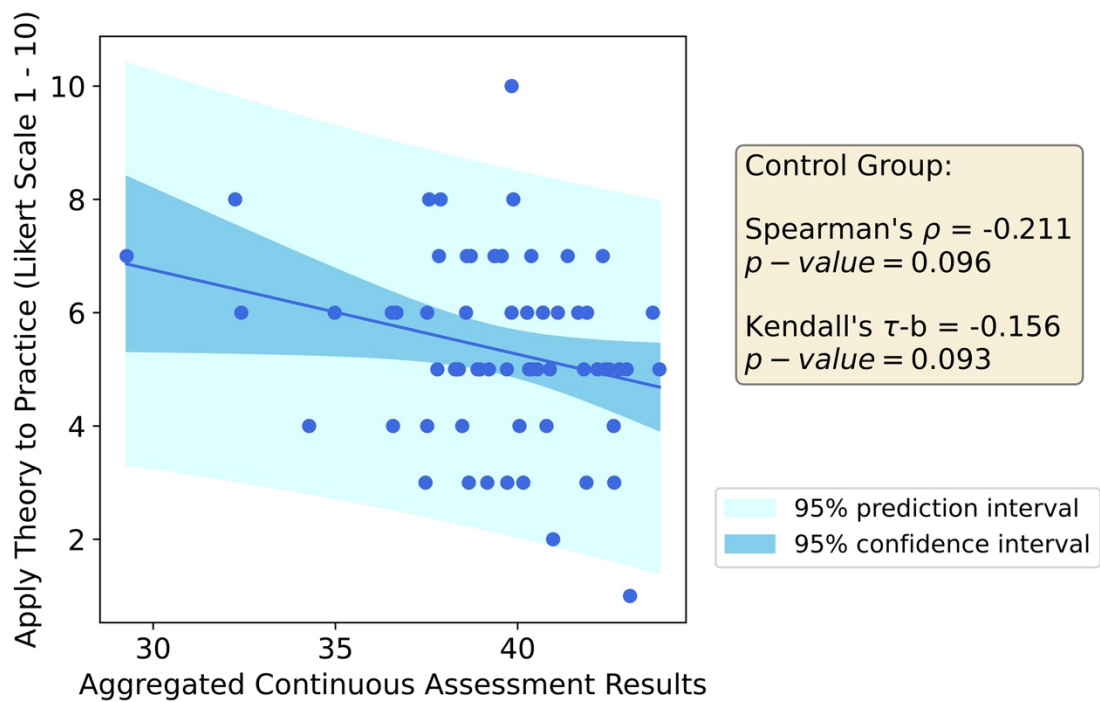


Figure 4. To determine to what extent the students have applied knowledge/skills learned from class to practice, students are asked to rate their opinion from 1 (almost nothing) to 10 (almost everything). 116 students participated in this anonymous survey.



(a) Experimental Group



(b) Control Group

Figure 5. Relationship between how much the students apply knowledge to practice and the aggregated continuous assessment results (including the results from the implementation of the system, the project design document and the project development journal), determined by Spearman's rank-order and Kendall's τ -b for both the Experimental group with $N = 36$ and the Control group with $N = 63$.

Table 1. Summary of the reasons why students choose different projects.

What are the main reasons for you and your team to take the competition project?	
Response ($N = 46$)	Percentage
I can take the opportunity to learn more.	89.1%
I want to practice developing real-world software projects.	78.3%
I am planning to work in software industry, doing this project can help me to be ready for jobs.	63.0%
It will help improve my academic profile.	41.3%
Others	10.9%
Why have you and your team decided to do the non-competition project?	
Response ($N = 70$)	Percentage
It is too difficult for us.	42.9%
We're afraid that if we couldn't do it well, it would affect our overall exam results.	42.9%
We feel we don't have the time to learn more from outside of the class.	28.6%
We are not interested.	21.4%
Others.	21.4%
Note: 116 students participated in the anonymous online survey.	

Table 2. Group descriptive statistics on the test results and the test of difference using the Independent Samples t-Test regarding students' knowledge and skill competency.

Group	N	Test Result			t -test		
		Mean	SD	SE	t	df	p
Control (non-competition project)	86	7.227	3.123	0.337	-0.424	145	0.672
Experimental (competition project)	61	7.442	2.925	0.374			

Note: SD - standard deviation, SE - standard error of mean, df - degree of freedom.

Table 3. Group descriptive statistics, test of equality of variances (Levene's) for students' coding hours, reading hours and meeting frequencies, and the comparison of difference for students' coding hours, reading hours and meeting frequencies of the two groups using the Welch *t*-test and Independent Samples *t*-test, respectively.

	Group	Descriptive Stat.			Levene's Test		<i>t</i> -Test			Effect Size Cohen's <i>d</i>
		N	Mean	SD	F	<i>p</i>	<i>t</i>	df	<i>p</i>	
Coding Hour	Exp	46	6.24	4.95	10.58	0.002	2.14*	58.37	0.018	0.43
	Ctl	70	4.57	2.34						
Reading Hour	Exp	46	7.5	5.72	23.06	0.0001	3.48*	61.34	0.0001	0.7
	Ctl	70	4.31	2.99						
Meeting Hour	Exp	46	2.03	1.01	1.49	0.226	-0.304**	104.7	0.762	
	Ctl	70	2.09	1.15						

Note: Exp - Experimental Group, Ctl - Control Group; * indicates the use of the Welch *t*-test, ** indicates the use of the Independent Samples *t*-test.

Table 4. Group descriptive statistics and Welch *t*-test for student's understanding of the subject matter and extension of their knowledge network.

Group	Descriptive Statistics			Welch <i>t</i> -test			Effect Size Cohen's <i>d</i>
	N	Mean	SD	<i>t</i>	df	<i>p</i>	
Control	86	2.89	1.85	-2.135	105.47	0.035	-0.366
Experimental	61	3.69	2.47				

Note: Levene's test is significant ($p = 0.005$), suggesting a violation of the equal variance assumption, thus the Welch *t*-test is used.

Table 5. Group descriptive statistics on the students' rating on application of knowledge/skills to practice.

Group	<i>N</i>	Mean	Median	SD	SE	Mean Rank
Control Group	63	5.35	5.0	1.62	0.2	5.0
Experimental Group	36	6.11	6.5	2.01	0.34	6.5

Mean ranks are calculated for the Mann Whitney test in Table 6.

Table 6. Comparison of intention to apply knowledge/skills learned from class to practice between the group of students working on the competition project and the group of students working on the non-competition project using Mann Whitney.

Test	Statistic	<i>df</i>	<i>p</i>	Effect Size
Independent Samples <i>t</i> -test	-2.03	97.0	0.045	-0.425
Mann-Whitney	856.5		0.041	-0.245

Note: For the Independent Samples *t*-test, the effect size is given by Cohen's *d*; for the Mann Whitney test, the effective size is given by the rank biserial correlation.

Table 7. Group descriptive statistics on the aggregated result from continuous assessments including the results from the implementation of the system, the project design document and the project development journal.

Group	<i>N</i>	Mean	Median	SD	max	min
Control Group	86	39.44	39.72	2.75	43.89	29.27
Experimental Group	61	41.31	41.06	2.36	45.12	34.18