

Students' Communication Self-efficacy and Its Impact on the Enhancement of Communication Skills in Software Engineering Project Courses

Chamikorn Hiranrat^{1,2}, Atichart Harncharnchai¹, Chompunoot Duangjan³

¹College of Arts, Media and Technology, Chiang Mai University, Chiang Mai, Thailand

²College of Computing, Prince of Songkla University Phuket Campus, Phuket, Thailand

³Faculty of Liberal Arts and Management Sciences, Prince of Songkla University
Surat Thani Campus, Surat Thani, Thailand

Abstract – Developing the communication skills of software engineering graduates to meet industry requirements is a challenge for educators. This study presents a project-based learning framework that promotes students' communication skills in a software engineering project course. The questionnaire on self-efficacy for software development (CSESD) was designed for students' self-assessment of their confidence in communication skills. Findings indicate that students' CSESD increased significantly after the course ended. Educators can apply the designed framework to software development-related project courses. The CSESD questionnaire can be used to assess students' confidence in their communication skills and assist educators in preparing students' readiness before graduation.

Keywords – Communication, self-efficacy, project-based learning, software engineering, PBL.

DOI: 10.18421/TEM123-50

<https://doi.org/10.18421/TEM123-50>

Corresponding author: Chamikorn Hiranrat,
College of Arts, Media and Technology, Chiang
Mai University, Chiang Mai, Thailand


Email: jamikorn.hi@phuket.psu.ac.th

Received: 24 March 2023.

Revised: 10 July 2023.

Accepted: 19 July 2023.

Published: 28 August 2023.

 © 2023 Chamikorn Hiranrat, Atichart Harncharnchai & Chompunoot Duangjan; published by UIKTEN. This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 4.0 License.

The article is published with Open Access at
<https://www.temjournal.com/>

1. Introduction

Software development is a knowledge-intensive process in which interaction, collaboration, and information sharing among team members occur throughout the development process [1].

With a strong emphasis on customer involvement, agile development approaches such as Scrum have been used as a software development over the past few years. Several activities, including requirement engineering, analysis, design, development, testing, implementation, and project management, involve intensive communication via documentation, meetings, discussions, and presentations [2], [3]. In software development, technical knowledge and skills are important, but the inclusion of soft skills is essential for being professional. Previous research indicated that soft skills, including motivation, commitment, teamwork, and communication, are required from graduates in software engineering (SE) and information technology [4], [5], [6].

During study in universities, students have practiced and developed their professional skills in software project development. However, previous surveys show that the communication skills of SE graduates do not meet industry expectations [4], [7], [8]. Student communication competence is related to career development and communication self-efficacy [9], [10], [11]. In self-efficacy theory, the belief in one's capability to perform a specific task is known as self-efficacy [12], [13]. Performance achievement is a source of self-efficacy based on personal mastery experience and impacts behavior change [12]. The achievement of project development assists students in enhancing their mastery of software development. The belief in communicating abilities when performing software development tasks was defined as communication self-efficacy for software development [14].

Communication success while developing a project utilizing abilities such as listening, discussing, and interviewing clients could enhance students' communication confidence. In computing education, working on a software project helps students develop both technical skills, such as how to use methods and tools for software development, and soft skills, such as problem-solving, teamwork, and communication skills [15], [16]. The traditional methodology, such as Waterfall model help students understand and acquire skill throughout the software development life cycle (SDLC). A modern software development approach, Agile, has been introduced, and most Agile methods such as Scrum and Extreme Programming, are widely used in the software industry [17]. The Scrum method [18] is suitable for PBL since Scrum emphasizes people and collaborative activities among teams and encourages students' self-learning, communication, and other soft skills [2], [19], [20].

Project-based learning (PBL) is commonly used in higher education to develop employability skills [21]. The PBL environment encourages students to combine technical and general skills through teamwork, information sharing, and communication to solve a problem case or scenario [2]. Applying PBL in the SE project course, students work in groups to develop a software project. They are assigned roles as project manager, systems analyst, programmer, and tester. Students could enhance their communication skills, such as active listening and communicating with clients or users to gather requirements and design solutions; written communication, such as SRS documents, manuals, and test reports, is also typically used [22]. Oral communication skills in discussions, meetings, and presentations are also necessary to exchange information among the team members throughout the software development process [23]. Effective communication allows timely feedback, facilitates fast and correct decision-making, and transfers team expertise [24]. Most of the tasks in Scrum require communication among team members and stakeholders. Therefore, students could gradually develop their communication skills and increase their confidence in communication through the software process.

This study applied PBL for software development using the Scrum method in a SE project course, with an emphasis on communication activities. An instrument for measuring students' CSESD was developed to measure students' confidence in their communication self-efficacy for software development.

While previous research has demonstrated that PBL improves students' communication abilities, the measurement of communication self-efficacy specifically for software development has been rare. Surveys were conducted before and after the SE project course to assess changes in students' CSESD. Furthermore, the result was compared with the survey conducted in the SE project course that applied a traditional software development process.

2. The SE Project Course Implemented the PBL for Software Development Using Scrum

The "Project in Software Engineering" course is a mandatory course in the undergraduate SE program at the College of Computing, Prince of Songkla University Phuket Campus in Thailand. This course focuses on developing software projects throughout the SDLC, and a traditional approach, the Waterfall model, has been used. However, the agile approach was more prevalent in software companies. Scrum is one of the agile methodologies that emphasizes team collaboration and requires extensive communication between team members and stakeholders. Therefore, the program committee has approved the use of Scrum in the project course for the 2020 academic year in order to prepare students for employment in the software industry after graduation.

The PBL framework for software development using Scrum of the SE project course was designed to improve students' communication skills in a senior-level SE project course. As illustrated in Figure 1, the course includes three essential components: the software development process, learning outcomes, and learning activities.

2.1. Learning Outcomes

Learning outcomes in the SE curriculum are used to evaluate whether students have acquired the course's information and skills. The SE program curriculum identifies the learning outcomes of the SE project course, which are listed in Table 1. Tasks related to communication abilities in the software development process were grouped to assess communication self-efficacy for software development. Students' confidence in their software development and communication skills, as well as their ability to accomplish learning outcomes, increases gradually as they complete tasks and milestones throughout the project's development.

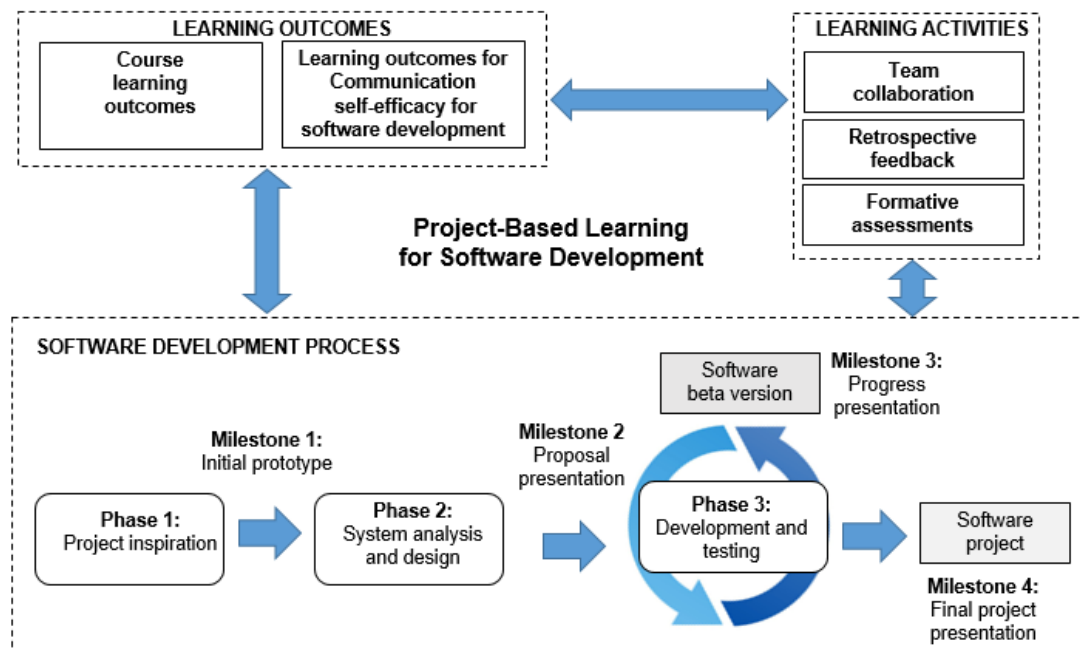


Figure 1. The PBL framework for software development of the SE project course

Table 1. Learning outcomes

Item	Description
LO1	Criticize software engineering practices, methods, tools, and techniques used in software project development and why they were selected.
LO2	Apply appropriate software engineering processes, methods, and tools in developing and managing a software project.
LO3	Develop a small software development project from a real-world problem in a small group within a given time frame.
LO4	Develop deliverables and artifacts in the software process through successful requirement engineering, design, development, testing, maintenance, and evaluation such as a project plan, SRS documents, prototypes, test documents, product manuals, and software product versions.
LO5	Communicate effectively in writing and orally with peers, advisors, clients, and stakeholders through a software process through interviews, discussions, presentations, and documents.
LO6	Develop life-long learning ability through continual reflection on the software development life cycle and teamwork procedures throughout the semester.
LO7	Manage learning and personal development, including time management, and organizational skill development.
LO8	Apply information and communication technologies and techniques to search, evaluate, and use scientific and technical information in order to achieve project goals.
LO9	Demonstrate professional, ethical, legal, security, and society-related issues with responsibilities.

2.2. The Software Development Process

To simulate the software development process in a near-real-world project, the software development process was constructed based on Software Development Life Cycle (SDLC) and Scrum practices. In this course, the software development process consists of three phases: project inspiration phase, systems analysis and design phase, and development and testing phase. Each phase includes activities to help students attain the course’s learning outcomes.

Phase 1: Project inspiration. Students are encouraged to select a topic from their areas of interest and understand the project’s significance. During this phase, students investigate the problems, interview users and stakeholders to understand the causes of problems, define users’ problems clearly, and brainstorm relevant solutions. Students may choose one or more solutions to suggest to customers. Then, students construct the initial prototype and solicit user feedback to confirm that the proposed solution effectively addresses user issues. Storyboards are used to illustrate the project’s objectives. The initial prototype is developed for proof of concepts which is the milestone of this phase.

Phase 2: System analysis and design. Students capture user requirements in more detail and create user stories, which are informal explanations of software requirements from the end users’ perspectives. Students categorize and prioritize software features based on user stories and estimate development time and time to acquire new technical knowledge and skills.

Students then draft the system design and architecture and write the Software Requirements Specification (SRS) documents, a project plan, and a project proposal. Students set up the project development environment and configure software for team collaboration and management. Students develop the operational prototype, submit all required documents, and present their work in the proposal presentation, which is the milestone of this phase.

Phase 3: Development and testing. Students develop a sprint plan, test specifications, and a test plan from user stories. Students set sprint goals, develop software functions, test, get user feedback, deliver artifacts according to the plan, and perform sprint reviews at the end of each sprint. The software beta version should be finished after 3-4 sprints. Students then present their progress to the committee by demonstrating the beta version of the software project with some features and deliverables according to the project plan. The progress presentation is the third milestone of the SE project course. Students then continue developing the software according to the plans and conduct usability and acceptance testing, including documents required by each phase: software specification, system design, test results, and manuals. All documents are compiled into the project report and submitted to the committee. The fourth milestone is the final project presentation.

2.3. Learning Activities

Learning activities are designed to foster students' communication for software development, including team collaboration, retrospective feedback, and formative assessments.

Team collaboration. Students formed groups of two or three depending on the scale of the project to enhance students' communication skills. In the *Project inspiration phase*, students in each team interview stakeholders to understand the problem, define the project's goals, and draft a prototype for testing with users. During the *System analysis and design phase*, students are assigned roles and responsibilities as business analysts, systems analysts, user interface designers, and project managers to prepare the required documents and operational prototypes for the proposal presentation. In the *Development and testing phase*, group collaboration simulates the Scrum team, consisting of a product owner, a Scrum master, and developers [18]. In each sprint, the product owner and Scrum master roles are rotated between team members. The product owner is responsible for assigning priorities to product backlog items and monitoring their status during sprints.

The Scrum master supports team members and assists in conflict resolution. Everyone is assigned the developer role and is responsible for developing software and delivering artifacts in accordance with sprint plans. Working with a team in the PBL environment encourages students to communicate orally and in writing with other members.

Retrospective feedback. The objective of the sprint retrospective is to develop solutions for enhancing quality and productivity. In each sprint, The Scrum team reports on the events of the previous sprint, offers their evaluation and feedback, and considers what went well, what concerns to emphasize, and how to handle them. The most significant enhancements to increase its efficiency are addressed as soon as possible and adapted in the next sprint [25], [26]. Continuous feedback from sprint retrospectives promotes team performance and customer satisfaction [27]. Students have a sprint-ending retrospective meeting with their advisor during the Development and Testing phase, which consists of three to four sprints. The purpose of the meeting is to discuss issues influencing the project schedule and share what students have learned from each sprint. Students report their work and reflect on their successes and failures to team members. Students receive feedback from their teams and discuss the difficulties and their solutions. Then, based on the difficulty and complexity of the tasks in the backlog, students decide on the goals and estimate the duration of the next sprint. Students are also encouraged to record their learning in an online logbook so that they can track their progress. The sprint retrospective creates a team learning atmosphere. The sprint retrospective feedback facilitates students' self-learning and knowledge-sharing with others. Students could develop communication skills such as active listening, discussions, and presenting technical knowledge.

Formative assessments. The formative assessment is acquiring information on student learning throughout a course. This information is then used to inform teaching and learning improvements [28]. Students have repeated opportunities to get feedback on the project's activities, tasks, and milestones throughout the semester [19]. Students can learn from the feedback and continue to revise their work until they achieve project goals. The feedback process of the formative assessments is vital for effective self-directed learning and for enhancing students' professional skills. The purpose of formative assessments is to assist students in developing self-awareness and self-discipline and enhancing their learning to meet course objectives [29].

Evaluating students' achievement using rubrics is based on course activities and software process deliverables, including weekly meetings with advisers, sprint retrospectives, presentations, and document writing.

Then, the course's schedule was developed by integrating the software development process, learning outcomes, and learning activities into the PBL environment.

2.4. The Course' Schedule

The 17-week course was worth three credits. Collaboration with users and teams to build small software project with deliverables and documentation needed 135 hours of student effort. Students were required to complete in mandatory courses such as requirement engineering, systems design, programming, software configuration, software project management, and software testing before registering the project course. The project course did not include formal lectures. Table 2 presents the SE project course's schedule.

Table 2. Course' schedule

Week	Activities
1	Workshop 1: Project inspiration (3hr.)
2	Workshop 2: Agile development: Scrum (3hr.)
3	Prepare project proposal.
4	Present project proposal.
5-7	Develop and test.
8	Progress presentation.
9-14	Develop and test.
15	Project presentation in the university annual project showcase exhibition.
16	Submit project reports.
17	Present the final project.

Course activities arranged in the schedule related to team collaboration, retrospective feedback, and formative assessment activities throughout the course. Students had to attend two workshops, including "Workshop 1: Project inspiration" and "Workshop 2: Agile development: Scrum" in the first two weeks of the semester.

In the first workshop, course objectives, the course schedule, activities, and expected learning outcomes were presented to students. Students formed groups of two or three and conducted research on businesses or challenges that interested them. Students then discussed with their team and selected a theme for which they would create a software system. Students were assigned to interview target users or stakeholders related to the software project and review the necessary literature or technology for software development.

Students decided on the project topic and consulted with SE lecturers regarding the project's scope and approval.

In the second week, the second workshop was held to introduce the agile development approach and Scrum. Students then collected and defined user requirements and created a storyboard and sketched a paper prototype to validate user requirements. Students conducted user testing and revised prototypes until acceptance was achieved. Students drafted the project proposal during the third week. Students produced user stories, estimated duration, and prioritized them in order of importance. Then, the system requirement specifications (SRS) document, the system architecture, and the system design were developed. Students configured the system development environment to develop an operational prototype. The project proposal and all documents were submitted to the committee. On the fourth week, students presented the proposal to the committee for their approval.

Students worked with the team throughout the fifth to seventh week to develop the software project by creating a sprint plan and prioritizing the product backlog. During sprint planning sessions, a student was assigned the position of Scrum master, whereas the adviser acted as product owner. According to the plan, students developed the software project, submitted code in a version control system, tested, and produced deliverables. A sprint review meeting and a retrospective were held at the ending of each sprint to obtain feedback from the team and the adviser.

In week eight, students demonstrate their software development progress to the committee. The groups that had not developed software in accordance with the plan were considered for a project plan review and scope adjustment. Until the fourteenth week, students developed their software project in accordance with the sprint plan and participated in sprint reviews and retrospectives with the adviser. In the week fifteen, students were given the opportunity to present their project to software industry recruiters and developers. Students submitted the final report in week sixteen and presented the final project in the last week of the semester.

The SE project course's activities required students to collaborate with their teams throughout the software development process. Students received feedback from peers, advisers, committees, and software industry professionals. Throughout the semester, retrospective feedback and formative assessments happened during Sprint review meetings, and presentations. These activities assist students in acquiring software engineering knowledge and communication skills necessary for software development.

3. Research Method

A project course implementing Scrum was designed and implemented on the experimental group in order to examine the effects of CSESD on the enhancement of communication skills in software development. To conduct a pretest and posttest for the experiment, the CSESD self-assessment questionnaire was developed. This study applied the quasi-experimental research design. A one-group pretest-posttest design was applied to the experimental group to examine the differences in students' CSESD before and after the study. Then, a non-equivalent control group design was employed to examine the CSESD difference after studying the project course between the experimental and control groups, which applied the traditional software development process. Descriptive statistics and the t-test analysis were used to examine the results.

3.1. Instrument Development

The 32 items of communication self-efficacy for software were collected from the previous literature [30], [31], including communication skills required for SE graduates and entry-level software development jobs. The items relating to oral and written communication abilities in general and technical software development contexts were measured on a 7-point Likert scale ranging from 1 (not confident at all) to 7 (absolutely confident), with the respondent being instructed to "Please rate your level of confidence (even if you have never done it before) in your ability to" The questionnaire was translated into Thai by an English-fluent professor and back into English by another professor. We have produced the questionnaires in two languages, Thai and English, which have been reviewed by three professionals who have taught software engineering courses for at least five years to ensure their accuracy and comprehension.

The content validity of the questionnaire was examined by a committee consisting of two SE professors and a software industry professional with more than five-year experience. An index of item-objective congruence (IOC) was evaluated using the content validity. Eight items were eliminated. Internal consistency was determined using a total of 24 valid items, each consisting of 12 oral and 12 written communication items, as shown in Table 3.

Table 3. Items of CSESD questionnaire

No.	Item
O1	Listen to others and consider their thoughts.
O2	Communicate to an audience from other countries or cultures.
O3	Explain precisely and accurately.
O4	Be nice to others, through words and tone.
O5	Develop the flexibility to communicate in different roles within an organization.
O6	During discussion, treat others with respect (e.g., when giving an opinion, debating potential solutions).
O7	Interview customers to gather requirements
O8	Interact with customers in prototyping user experience and design ideas.
O9	Discuss and review of plans, processes, tools, and issues with development team.
O10	Present technical information to groups and solicit ideas is required to get feedback.
O11	Communicate via formal presentations to groups.
O12	Communicate via informal presentations to a group
W1	Gathering information, summarizing, and simplifying to others for decision making.
W2	Communicate via visuals (e.g., figures and tables).
W3	Use e-mail and instant messaging appropriately (e.g., read before sending, know when to talk in person).
W4	Write formal documents, and use correct terminology, spelling, and grammar most of the time.
W5	Write in English fluently.
W6	Capture user requirements and notate with user stories.
W7	Write formal requirements/specifications.
W8	Craft scenarios, storyboards, information architectures, features and interfaces.
W9	Write detailed programming specifications after analyzing business requirements for system subcomponents.
W10	Communicate via code comments and check-in notes.
W11	Write and organize the source code for reading and comprehending easily to modify, extend, or rewrite software easily.
W12	Produce test specifications, test plan, test manuals, and test results required writing skills.

The questionnaire was tried out with thirty senior students to determine the reliability of the questionnaire items. Overall, the questionnaire's Cronbach's alpha coefficient was 0.953.

The computed alpha values for self-efficacy in oral and written communication were 0.93 and 0.91, respectively. According to Cronbach (1951), Cronbach’s alpha for all items surpassing 0.70 indicated the acceptability of the questionnaire [32].

3.2. Data Collection and Participants

Data collection was performed in the software engineering program at a university in Thailand. The control group consisted of seniors majoring in SE who enrolled in the SE project course during the first semester of 2019 and developed their projects using a traditional software approach, the Waterfall model. At the end of the semester, students were requested to complete the CSESD questionnaire as a posttest or post-study. The experimental group consisted of senior SE students enrolled in the SE project course during the first semester of the 2020 academic year. Students were asked to complete the CSESD questionnaire before and after studying the project course.

According to the curriculum requirements, all students had already enrolled in mandatory SE program courses, such as Requirement Engineering and System Modeling, Software Configuration Management, Software Verification and Validation, and Software Project Management. Students formed groups of three to develop small software projects. Seven lecturers in the SE department served as project advisors and committee members to assess students’ learning accomplishment.

Participants were between the ages of 21 and 23. In 2019, 35 students enrolled in the project course, and 27 of them (or 77.1%) completed the questionnaires. Seventeen students (63%) were male, and ten (37%) were female. A total of 33 students registered for the project course in the 2020 academic year, and 31 of them (93.9%) completed the surveys. Twenty-three (74%) of the students were male, while eight (26%) were female.

4. Data Analysis and Results

Data from CSESD questionnaires were analyzed using SPSS version 26. Table 4 provides descriptive statistics of means and standard deviations for the control group (post-study) and the experimental group (pre-study and post-study). The graph in Figure 2 illustrated the means of CSESD items.

According to Table 4, comparing means of CSESD items between pre-study and post-study for the experimental group, item O6 (“During discussion, treat others with respect (e.g., when giving an opinion, debating potential solutions).”) was rated as having the highest confidence level (5.35) in the pre-study, whereas item O1 was rated as having the highest confidence level (6.0) in the post-study.

While item W5 (“Write in English fluently.”) earned the lowest rating score in both pre-study and post-study (3.52 and 4.06, respectively). Comparing the post-study CSESD item means of the control and experimental groups reveals that the confidence level of item O1 (“Listen to others and consider their thoughts.”) was rated as the highest by both groups (5.70 and 6.00), while the confidence level of item W5 (“Write in English fluently.”) was rated as the lowest (4.07 and 4.06).

Table 4. Means and standard deviations in CSESD items in control and experimental groups

Item	Control Gr. (N=27)		Experimental Gr. (N=31)			
	Post-study		Pre-study		Post-study	
	Mean	Std.	Mean	Std.	Mean	Std.
O1	5.70	1.137	5.26	1.154	6.00	0.816
O2	4.33	1.074	3.87	0.991	4.26	1.237
O3	4.33	1.177	4.06	0.854	4.61	0.761
O4	5.19	1.331	5.13	1.176	5.48	1.061
O5	4.96	1.255	5.03	1.140	5.39	0.844
O6	5.04	1.192	5.35	1.082	5.77	0.884
O7	4.85	1.199	4.45	1.060	5.16	1.036
O8	5.15	1.231	4.81	1.167	5.39	0.803
O9	4.89	1.155	4.84	1.036	5.16	0.934
O10	4.89	1.251	4.68	1.013	5.19	0.792
O11	4.85	1.134	4.68	1.107	5.29	0.938
O12	4.85	1.134	4.68	1.107	5.29	0.938
W1	4.93	1.269	4.71	1.039	5.19	0.749
W2	5.19	1.075	5.16	1.036	5.45	0.888
W3	5.15	1.064	4.84	1.157	5.52	0.851
W4	5.00	1.177	4.65	1.279	5.06	1.153
W5	4.07	1.385	3.52	1.262	4.06	1.413
W6	5.07	1.174	4.42	1.057	5.00	0.894
W7	4.96	1.192	4.45	0.961	5.45	0.810
W8	4.89	1.311	4.45	0.850	5.16	0.898
W9	4.56	1.251	4.26	0.815	5.00	0.730
W10	4.63	1.275	4.23	0.884	5.06	1.209
W11	4.44	1.251	4.10	1.136	4.87	0.846
W12	4.33	1.271	4.19	1.167	4.94	1.124

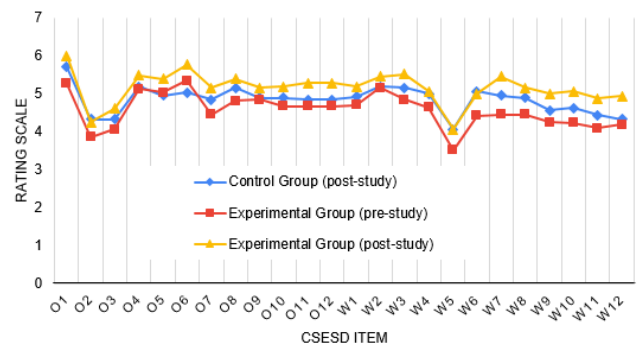


Figure 2. Means of CSESD items

Table 5 displays the pair sample t-test for pre-study and post-study CSESD items for the experimental group.

The significance probability (p-value) of the majority of items was below than the significance level (0.05), indicating that there was a statistically significant difference between the pre-study and post-study of the CSESD results except for the item O2 (“Communicate to an audience of persons from other countries or cultures.”), O4 (“Be nice to others, through words and tone.”), O5 (“Develop the flexibility to communicate in different roles within an organization.”), O6 (“During discussion, treat others with respect (e.g., when giving an opinion, debating potential solutions).”), O9 (“Discuss and review of plan, process, tools, and issues with development team.”), and W2 (“Communicate via visuals (e.g., figures and tables).”). The t-test reveals no statistically significant difference between the pre-study and post-study means for these six CSESD items, despite the fact that the post-study means are higher.

Table 5. The test result of the pair sample t-test on the pre-study and post-study of CSESD items of the experimental group

Item	t	Sig.(2-tailed)	Item	t	Sig.(2-tailed)
O1	3.268	.003	W1	2.540	.016
O2	1.931	.063	W2	1.393	.174
O3	2.882	.007	W3	3.087	.004
O4	2.006	.054	W4	7.905	.000
O5	1.827	.078	W5	5.343	.000
O6	1.938	.062	W6	2.568	.015
O7	3.406	.002	W7	4.947	.000
O8	2.816	.009	W8	4.383	.000
O9	1.541	.134	W9	4.004	.000
O10	2.633	.013	W10	4.055	.000
O11	3.058	.005	W11	4.353	.000
O12	3.058	.005	W12	3.338	.002

N = 31, df = 30, p < .05

The CSESD items were divided into oral and written communication groups, item O1-O12 and item W1-W12, respectively. The means and standard deviations of the oral, written, and overall CSESD are presented in Table 6. The experimental groups’ post-study oral, written, and total CSESD ratings were greater than the pre-study ratings. Comparing the post-study means between the experimental and control groups, the experimental group reported greater levels of oral, written, and overall CSESD confidence.

The paired sample t-test on the experimental group was conducted. Table 7 reveals significant differences of 0.000 (p < .05), indicating a statistically significant difference between pre-study and post-study oral, written, and overall CSESD scores. The finding demonstrates that generally, students’ confidence in their communication skills for software development increased after completing a project in the SE project course using Scrum.

Table 6. Means and standard deviations categorized by communication types

Group	Measurement	Category	Mean	Std.
Control	Post-study	Oral	4.92	0.93
		Written	4.77	0.92
		Overall	4.84	0.87
Experimental	Pre-study	Oral	4.74	0.83
		Written	4.41	0.79
		Overall	4.58	0.77
	Post-study	Oral	5.25	0.52
		Written	5.06	0.48
		Overall	5.16	0.45

Table 7. The test result of the pair sample t-test on the pre-study and post-study of CSESD of the experimental group

	Mean	Std.	t	Sig.(2-tailed)
Oral	0.51	0.66	4.31	0.00
Written	0.65	0.69	5.23	0.00
Overall	0.58	0.65	5.00	0.00

N = 31, df = 30, p < .05

The t-test on independent samples was used to assess the mean difference between the control and experimental groups. The result in Table 8 shows that the significance levels of oral, written, and overall CSESD were 0.111, 0.141, and 0.102, respectively. A p-value greater than 0.05 indicates that the means of CSESD in control and experimental groups are not significantly different.

Table 8. Test result of the independent sample t-test on CSESD post-study of the control and experimental groups

	Levene’s Test for Equality of Variances		t-test for Equality of Means		
	F	Sig.	t	Sig. (2-tailed)	Std. Error
Oral	7.322	0.009	1.629	0.111	0.20
Written	4.613	0.036	1.503	0.141	0.20
Overall	5.323	0.025	1.677	0.102	0.19

p < .05

According to the results, the experimental group acquired confidence in their software development communication skills. However, there is room for improvement. Students working with their classmates may not gain confidence in their oral communication skills in software development, as presented in Table 5. The results may differ if students build a real-world project or interact directly with real clients. In addition, the finding in Table 4 suggests that the faculty should provide strategies for enhancing students’ English writing skills to increase their confidence when graduating.

Comparing the post-study CSESD means of the control and experimental groups, the results in Table 8 indicate no statistically significant differences. However, Table 6 shows that students in the experimental group, who applied the Scrum methodology, rated their confidence in communication skills higher than those in the control group that used the Waterfall model. Different software development processes implemented in the project course require different activities and communication tasks. Students in the control group developed a software project based on the Waterfall model with five primary phases: requirements, analysis, design, development, and testing. The performance evaluation focused on artifacts and reports generated in each phase, software product completion, and project presentations. The experimental group used Scrum, an incremental model that divides the process into sprints. Scrum requires additional engagement between team members and stakeholders during sprints for planning, sprint reviews, and retrospective meetings. In each sprint, students participated in activities and received continuous feedback for improvement in the next sprint. Students' communication skills could continuously improve throughout the development process, leading to enhancing their CSESD.

5. Conclusion

In this study, a software engineering project course implementing the PBL for software development using Scrum was presented. The CSESD self-assessment instrument was developed to evaluate students' confidence in oral and written communication skills for software development. The results indicate that the course improves CSESD among course participants. This study has contributed to SE education research as it provides a guide for conducting PBL courses or training for software development project courses emphasizing communication activities. In addition, the CSESD questionnaire allows students to assess their confidence in their software development-related communication skills. Instructors can use the results from the self-assessment to guide the improvement of students' communication skills in which they lack confidence in order to prepare them for careers in SE following graduation.

The limitation of this study is the relatively small sample size in a single SE program. Each SE program has its own set of learning outcomes. Implementing the course in other SE programs or increasing the number of participants may yield different consequences. Future studies may focus on integrating the courses with industry partnerships.

Students can benefit from industry collaboration to improve CSESD by communicating with real clients and stakeholders. The complexity of real-world problems might challenge students to acquire knowledge and skills for solutions. Expanding the project scope with more complicated problems necessitates a larger team within a limited timeframe. We could break the project into sub-projects and employ two or three teams responsible for different functionalities. Students would collaborate with greater effort on integration, project management, and communication with clients and other teams. Based on student learning assessment data and industry feedback, we could improve the course to better prepare students for employment.

References:

- [1]. Giuffrida, R., & Dittrich, Y. (2015). A conceptual framework to study the role of communication through social software for coordination in globally distributed software teams. *Information and Software Technology*, 63, 11–30.
- [2]. Chassidim, H., Almog, D., & Mark, S. (2018). Fostering soft skills in project-oriented learning within an agile atmosphere. *European Journal of Engineering Education*, 43(4), 638–650.
- [3]. Kluender, J., Unger-Windeler, C., Kortum, F., & Schneider, K. (2017). Team meetings and their relevance for the software development process over time. *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 313–320.
- [4]. Garousi, V., Giray, G., Tuzun, E., Catal, C., & Felderer, M. (2020). Closing the gap between software engineering education and industrial needs. *IEEE Software*, 37(2), 68–77.
- [5]. Hiranrat, C., & Harncharnchai, A. (2018). Using text mining to discover skills demanded in software development jobs in Thailand. *Proceedings of the 2nd International Conference on Education and Multimedia Technology*, 112–116.
- [6]. Wang, X., Lin, X., & Hajli, N. (2019). Understanding software engineers' skill development in software development. *Journal of Computer Information Systems*, 61(2), 108–117.
- [7]. Stevens, M., & Norman, R. (2016). Industry expectations of soft skills in IT graduates: A regional survey. *Proceedings of the Australasian Computer Science Week Multiconference*, 1–9.
- [8]. Exter, M., Caskurlu, S., & Fernandez, T. (2018). Comparing computing professionals' perceptions of importance of skills and knowledge on the job and coverage in undergraduate experiences. *ACM Transactions on Computing Education*, 18(4), 1–29.
- [9]. Anderson, C. B., Lee, H. Y., Byars-Winston, A., Baldwin, C. D., Cameron, C., & Chang, S. (2016). Assessment of scientific communication self-efficacy, interest, and outcome expectations for career development in academic medicine. *Journal of Career Assessment*, 24(1), 182–196.

- [10]. Gaffney, A. L. H. (2011). Measuring students' self-efficacy for communication. *International Journal of Art & Design Education*, 30(2), 211–225.
- [11]. Song, Y., Yun, S. Y., Kim, S.A., Ahn, E.K., & Jung, M. S. (2015). Role of self-directed learning in communication competence and self-efficacy. *Journal of Nursing Education*, 54(10), 559–564.
- [12]. Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review*, 84(2), 191–215.
- [13]. Bandura, A. (1986). The explanatory and predictive scope of self-efficacy theory. *Journal of Social and Clinical Psychology*, 4(3), 359–373.
- [14]. Hiranrat, C., Harncharnchai, A., & Duangjan, C. (2021). Theory of planned behavior and the influence of communication self-efficacy on intention to pursue a software development career. *Journal of Information Systems Education*, 32(1), 40–52.
- [15]. Pérez-López, M. C., González-López, M. J., & Rodríguez-Ariza, L. (2019). Applying the social cognitive model of career self-management to the entrepreneurial career decision: The role of exploratory and coping adaptive behaviours. *Journal of Vocational Behavior*, 112, 255–269.
- [16]. Souza, M., Moreira, R., & Figueiredo, E. (2019). Students perception on the use of project-based learning in software engineering education. *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, 537–546.
- [17]. *15th Annual State of Agile Report*. (2021). Digital.ai. Retrieved from: <https://digital.ai/resource-center/analyst-reports/15th-state-of-agile-report/> [accessed: 02 February 2023].
- [18]. Schwaber, K., & Beedle, M. (2002). *Agile software development with Scrum*. Prentice Hall.
- [19]. Magana, A., Seah, Y. Y., & Thomas, P. (2018). Fostering cooperative learning with Scrum in a semi-capstone systems analysis and design course. *Journal of Information Systems Education*, 29(2), 75–92.
- [20]. Paasivaara, M., Vanhanen, J., & Lassenius, C. (2019). Collaborating with industrial customers in a capstone project course: The customers' perspective. *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 12–22.
- [21]. Fitzsimons, J., & Turner, R. (2013). Integrating project-based learning into an undergraduate programme using Web 2.0 and videoconferencing. *Journal of Applied Research in Higher Education*, 5(1), 129–140.
- [22]. Misnevs, B., & Demiray, U. (2017). The role of communication and meta-communication in software engineering with relation to human errors. *Procedia Engineering*, 178, 213–222.
- [23]. Ahmed, F., Fernando Capretz, L., Bouktif, S., & Campbell, P. (2012). Soft skills requirements in software development jobs: A cross-cultural empirical study. *Journal of Systems and Information Technology*, 14(1), 58–81.
- [24]. Bjarnason, E., Wnuk, K., & Regnell, B. (2011). Requirements are slipping through the gaps—a case study on causes & effects of communication gaps in large-scale software development. *2011 IEEE 19th International Requirements Engineering Conference*, 37–46.
- [25]. Andriyani, Y., Hoda, R., & Amor, R. (2017). Reflection in agile retrospectives. In *Agile Processes in Software Engineering and Extreme Programming: 18th International Conference, XP 2017, Cologne, Germany, May 22-26, 2017, Proceedings 18* (pp. 3-19). Springer International Publishing.
- [26]. Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide*. ScrumGuides. Retrieved from: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf> [accessed: 21 February 2023].
- [27]. Kortum, F., Klünder, J., & Schneider, K. (2019). Behavior-driven dynamics in agile development: The effect of fast feedback on teams. *2019 IEEE/ACM International Conference on Software and System Processes (ICSSP)*, 34–43.
- [28]. Cifrian, E., Andrés, A., Galán, B., & Viguri, J. R. (2020). Integration of different assessment approaches: Application to a project-based learning engineering course. *Education for Chemical Engineers*, 31, 62–75.
- [29]. Hassan, O. A. B. (2011). Learning theories and assessment methodologies – an engineering educational perspective. *European Journal of Engineering Education*, 36(4), 327–339.
- [30]. Ruff, S., & Carter, M. (2009). Communication learning outcomes from software engineering professionals: A basis for teaching communication in the engineering curriculum. *2009 39th IEEE Frontiers in Education Conference*, 1–6.
- [31]. Ahmad, M. O., Lenarduzzi, V., Oivo, M., & Taibi, D. (2018). Lessons learned on communication channels and practices in agile software development. *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*, 929–938.
- [32]. Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika*, 16(3), 297–334.