


## Article

# Effects of Flipped Learning Approaches on Students' Learning Performance in Software Engineering Education

Yen-Ting Lin 

Department of Computer Science, National Pingtung University, Pingtung City 900391, Taiwan; ricky014@gmail.com

**Abstract:** Software engineering education plays an important role in keeping students educated with software technologies, processes, and practices that are needed by industries. Nevertheless, the nature of software engineering learning activities in traditional classrooms is limited in scope and time, making it more difficult to achieve a proper balance between theory and practice and address industrial demands. This makes scant provision for assisting students in keeping their software engineering knowledge current. To support software engineering education, flipped learning is a suitable strategy. Prior studies have shown that students' perceptions in flipped learning environments are better than those in traditional learning environments. Nevertheless, in flipped learning, students may not have sufficient ability to conduct learning out of class. Therefore, the flipped learning strategy should aim to meet the needs of students to ensure that they get the appropriate support or feedback during the learning process before the class. The aim of this study was to propose a flipped learning diagnosis approach to promote students' learning out of class in the flipped classroom. To explore students' learning performance in software engineering courses, three classes of students were invited to learn with three different learning approaches (traditional learning approach, flipped learning approach, and flipped learning diagnosis approach). The results showed that the students who learned with the flipped learning diagnosis approach outperformed those students who learned with the flipped learning approach or the traditional learning approach.

**Keywords:** flipped classroom; software engineering; engineering education; learning performance; quality education



**Citation:** Lin, Y.-T. Effects of Flipped Learning Approaches on Students' Learning Performance in Software Engineering Education. *Sustainability* **2021**, *13*, 9849. <https://doi.org/10.3390/su13179849>

Academic Editors: Kris Law, Yuk-Ming Tang and Jukka Majava

Received: 3 August 2021

Accepted: 30 August 2021

Published: 2 September 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

To date, software engineering has changed significantly to advance the development of various types and scales of software products. In 1989, Ford and Gibbs firstly mentioned the gap between the software industry and software engineering education [1]. During the last three decades, although software engineering education continues to evolve, addressing industrial demands is still an open question for software engineering education [2]. One of the major challenges in software engineering education stems from the dual nature of the discipline of software engineering, which is the disciplines of computer science and engineering [3]. This has a direct impact on the amount of material that instructors are required to cover for both theoretical and practical learning activities and address industrial demands in software engineering classes. The nature of classroom learning activities is limited in scope and time, making it more difficult to achieve the aforementioned objectives [4]. Therefore, to teach software engineering, most instructors can only provide a theoretical-based or practical-based instruction to students in traditional learning environments. This makes scant provision for assisting students in keeping their software engineering knowledge current [2].

To support software engineering education, flipped learning is a suitable strategy for teachers and students. Flipped learning provides teachers' lectures in traditional classrooms as video clips for students to learn before the class. In class, teachers are able to encourage

students to engage in more high-order thinking learning activities when compared to traditional classroom settings [5]. McGuinness and Vlachopoulos (2019) indicated that student's perception is part of building a successful learning environment [6]. Prior studies have shown that students' perceptions (i.e., learning satisfaction, self-efficacy, and learning motivation) in flipped learning environments are better than those in traditional learning environments [7,8].

In flipped learning environments, Akçayır and Akçayır [9] indicated that students' learning performance out of class is important since it affects how instructors and students engage in learning activities in class. Nevertheless, out of class, students may not have sufficient ability to learn new knowledge and evaluate their learning results. Therefore, students could have a high risk of participating in learning activities on faulty foundations in class since they lack suitable learning supports before the class [10].

Therefore, the aim of this study is to adopt the flipped learning strategy to promote software engineering education. Moreover, a learning diagnosis system was proposed to support students' learning activities out of class in the flipped classroom. To explore students' learning performance in software engineering courses, in this study three classes of students were invited to learn with three different software engineering learning approaches (traditional learning approach, flipped learning approach, and flipped learning diagnosis approach), and their learning achievement, learning motivation, learning attitude, problem solving ability, and learning behaviors were further investigated.

## 2. Literature Review

### 2.1. Software Engineering Education

Different from physical object design, software is an abstract object and it is not limited by physical constraints. Therefore, during software development processes, various unpredictable problems could be encountered, particularly when developing complex and large-scale software [11]. To address the problems, software engineering is an effective and efficient way to support software developers in applying scientific procedures, methodologies, and models to develop software that can meet the needs of end-users [12,13]. As a result, software engineering education is important for software development [14,15].

Generally, to teach software engineering in higher education, teachers adopt the traditional teaching strategy to deliver theoretical concepts due to the limited course duration in university curriculum plans. Nevertheless, software engineering education involves a range of issues from software lifecycle to project management [16]. Moreover, to address the gaps between software industry and education, Mishra et al. indicated that software engineering education should also integrate some important issues, including interdisciplinary skills, practice experience, communication, skills on continuing education, and professionalism [17]. Obviously, the traditional teaching approach could not prepare students to stay current in the face of rapid change [18]. Therefore, previous studies suggested that software engineering education should rely on learner-centered strategies to enable students to practice their skills and interact with teachers and peers in relatively realistic learning environments [19–21].

One of the commonly used approaches is project-based learning (PjBL) [22,23]. Moreover, experiential-based learning is also a teaching strategy to support software engineering education [24]. Furthermore, Massive Open Online Courses (MOOCs) represents one commonly adopted strategy [25]. Beecham [15] indicated that, regardless of the teaching approach used, the relevance of high-order thinking skills obtained in software engineering education is important to teachers and students [15]. Cico et al. [2] also indicated that in addition to PjBL approaches in software engineering education, other potential learning approaches, i.e., flipped learning, still lack significant investigation [2].

### 2.2. Flipped Classroom

In the traditional classroom, teachers usually conduct lecture-based instructions during the course session and students receive the instructions passively. Furthermore, in

order to facilitate students' thinking, teachers typically assign exercises or practices as homework to students out of class. Nevertheless, to complete the assignments, students generally need to interact and discuss with peers and teachers to facilitate high-order thinking skills [26].

In 2011, Khan used the term "flipping the classroom" in his TED talk [27]. In the flipped classroom, the unidirectional delivery of teachers' instructions in the traditional classroom is generally changed on the web through digital materials and students can learn the instructions out of class before taking the class. In addition, the assignments out of class are changed to conduct in class and teachers and students can have high interactions and discussions with regard to the assignments to facilitate high-order thinking skills during the course session [28]. Based on Bloom taxonomy [29], teachers who adopt the flipped classroom pedagogy can arrange limited course sessions well to engage students in higher forms of thinking in class, such as applying and analyzing concepts, processes, procedures, and principles, rather than just remembering facts [30]. Moreover, through high interactions with teachers and peers in class, students are encouraged to facilitate their learning motivation, develop their thinking skills, promote their problem solving abilities, and enhance the learning outcomes [31–33]. Several studies also indicated that the flipped classroom pedagogy enhances students' learning motivation, maintains positive learning attitude, increases learning achievement, and facilitates engagement in the learning process [34–38].

In the flipped classroom, students' learning performance out of class is important since it affects how teachers and students engage in learning activities in class [9]. Nevertheless, out of class, students may not have sufficient ability to learn new knowledge and evaluate learning results. In this circumstance, students could have a high risk of participating in learning activities on faulty foundations in class since they lack suitable learning supports before the class [10,39].

Therefore, to promote software engineering education, the aim of the study is to adopt the flipped learning strategy in software engineering courses. Moreover, a flipped learning diagnosis system was proposed to improve students' learning out of class in the flipped classroom. To explore students' learning performance in software engineering courses, in this study three classes of students were invited to learn with three different software engineering learning approaches (traditional learning approach, flipped learning approach, and flipped learning diagnosis approach), and their learning achievement, learning motivation, learning attitude, problem solving ability, and learning behaviors were further investigated as the following research questions.

1. Is there any difference among different software engineering learning approaches in terms of the students' learning achievement?
2. Is there any difference among different software engineering learning approaches in terms of the students' learning motivation and learning attitude?
3. Is there any difference among different software engineering learning approaches in terms of the students' problem solving ability?
4. Is there any impact of the students' learning logs derived from the flipped learning diagnosis system on the students' learning achievement?

### 3. Flipped Learning Diagnosis System

In this study, a flipped learning diagnosis system was applied to support students' flipped learning before the class. The learning diagnostic methodology has been proposed and evidenced in different studies to support students' learning diagnosis activities [18,21,40–42].

Figure 1 shows the architecture of the flipped learning diagnosis system. The system was a responsive cross-platform web application. Teachers and students can use various learning devices to operate the functions of the system in flipped learning activities. The system was composed of two major subsystems—student learning and diagnosis system and teacher knowledge management system.

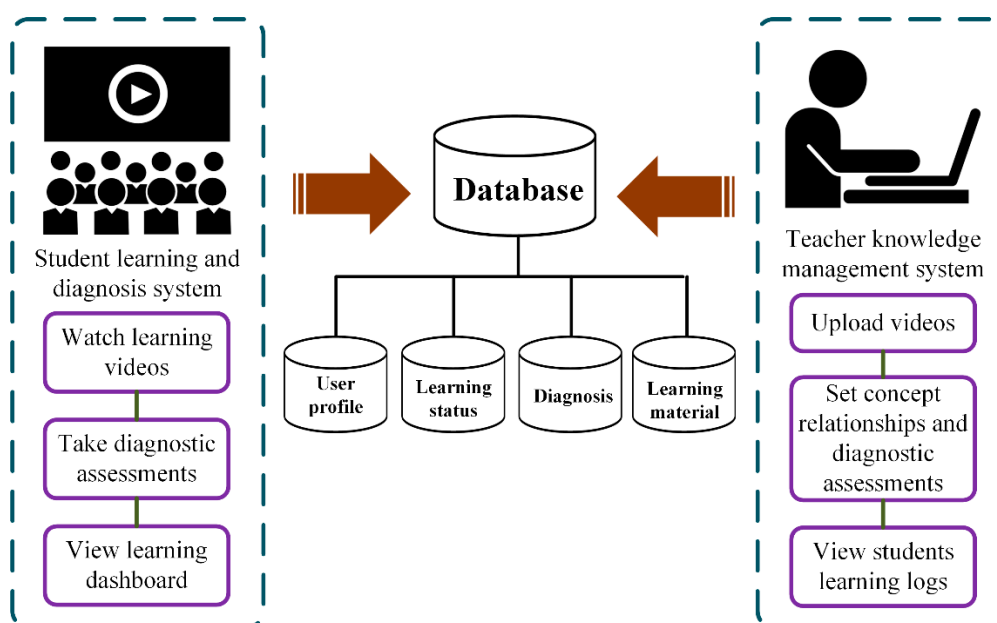


Figure 1. The architecture of the flipped learning diagnosis system.

To conduct flipped learning activities, teachers can use the knowledge management system to handle learning resources and assessments for software engineering courses, as shown in Figure 2. Teachers can manage course subjects, concepts, video clips, slides, and assessments. Moreover, to provide the diagnostic service in the system, teachers can assign the relationships between concepts and test items in an assessment, and the relationships between concepts.

**Concepts List (Top-Left Screenshot):**

#	Concepts
1	Software Processes
2	Waterfall Model
3	Incremental Development
4	Incremental Delivery
5	Agile Development

**Concept Relationships Matrix (Top-Right Screenshot):**

#	1	2	3	4	5	6	7	8	9
Concepts	Software Processes	Waterfall Model	Incremental Development	Incremental Delivery	Agile Development	Extreme Programming	Requirements Engineering	Functional Requirements	Software Requirements Document
1 Software Processes	1	0.8	0	0	0	0	0	0	0
2 Waterfall Model		1	0	0	0	0	0	0	0
3 Incremental Development			1	0.8	0	0	0	0	0
4 Incremental Delivery				1	0	0	0	0	0
5 Agile Development					1	0.8	0	0	0
6 Extreme Programming						1	0	0	0

**Items List (Bottom-Left Screenshot):**

#	Item	Answer	Other Choices
1	What is the model that breaks project activities into linear sequential phases?	(A). Waterfall Model	(B). Incremental Model (C). Fountain Model (D). Agile Development Model
2	What is the model called software lifecycle?	(A). Waterfall Model	(B). Incremental Model (C). Agile Development Model (D). Spiral Model

**Item-Concept Relationships Matrix (Bottom-Right Screenshot):**

#	1	2	3	4	5	6	7	8
Title	Software Processes	Waterfall Model	Incremental Development	Incremental Delivery	Agile Development	Extreme Programming	Requirements Engineering	Functional and Non-Functional Requirements
1 What is the model that breaks project activities into linear sequential phases?	0	1	0	0	0	0	0	0
2 What is the model called software lifecycle?	0.8	1	0	0	0	0	0	0

Figure 2. Snapshots of the knowledge management interfaces.

During flipped learning processes, students are able to use any learning device with a web browser to access the system on the Internet. Through the system, students can watch learning videos to learn the fundamental concepts to facilitate their prior knowledge before the class, as shown in Figure 3. Moreover, they can take assessments to evaluate and diagnose their learning results. As shown in Figure 4, the student took a diagnostic

assessment which involves three test items and five concepts assigned by the teacher. The relationship between the test items and the concepts and the relationship between the concepts were assigned by the teacher, as shown in Figure 3. After the students have taken the assessment, the system can obtain the relationship between the student's answers and the test items. Based on the relationships, the system can determine which concepts the student is weak in and provide recommended learning resources on the individual dashboard to improve their learning performance before the class. The detail illustration of the diagnostic methodology can be found in [40].

The demo of the proposed system was published on YouTube as the web link: <https://youtu.be/BQzb3S38TE0> (accessed on 31 August 2021). Personal information was blacked out in the video.

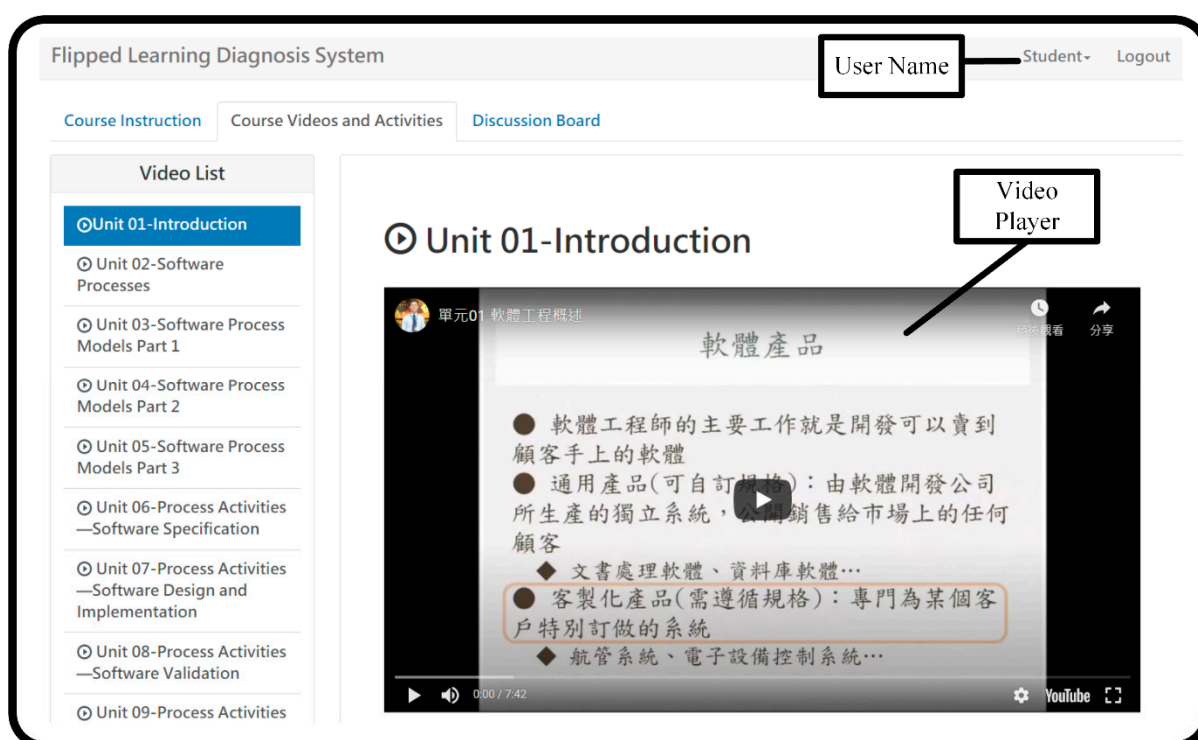


Figure 3. Screenshot of the student learning interface.

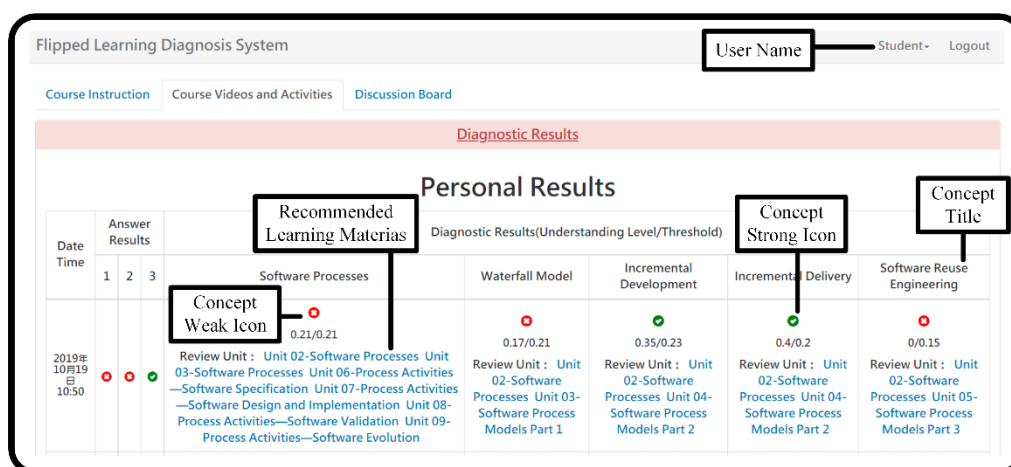


Figure 4. Screenshot of the diagnostic result interface.



## 4. Methodology

To investigate the differences among different software engineering learning approaches in terms of students' learning performance, a quasi-experiment was conducted in a Taiwanese university. The subject-matter area of software engineering in the experiment was focused on software lifecycle, including software processes, requirements engineering, design and implementation, software testing, and software evolution.

### 4.1. Participants

The participants were 54 students (ranging in age from 20 to 22) from three classes of the software engineering course, which is an elective subject of the department of computer science in the university. All students from the three classes were divided into three groups, respectively. The first group of 15 students served as the control group. The second group of 19 students served as the experimental group 1. The third group of 20 students served as the experimental group 2. The control group was supported by the traditional teaching strategy without the proposed system. The experimental group 1 was supported by the flipped learning strategy without the proposed system. The experimental group 2 was supported by the flipped learning strategy with the proposed system. All students had never learned software engineering before. The three classes were taught by the same instructor, who had more than 10 years of software engineering teaching experience.

### 4.2. Instrument

This study adopted measuring tools such as learning achievement tests, learning motivation, learning attitude, and problem solving ability questionnaires. Besides, the students' learning logs were also analyzed.

The pre-test was used to identify participants' prior knowledge of software engineering. The post-test was used to assess the learning performance after they completed the course. The pre-test and the post-test covered all relevant topics of the teaching unit and were administered as paper-and-pencil tests with a perfect score of 100. The items set in the tests were designed with the help of two instructors who had taught software engineering for over 10 years. The KR-20 reliability of the test was 0.715, indicating an acceptable internal consistency. The KR-20 reliability coefficient is used to estimate the internal consistency of an achievement test when a test measures a unidimensional trait and if it is scored dichotomously [43]. Moreover, the mean difficulty index ( $p$ -value) of the items was 0.53. The difficulty index describes the percentage of students who correctly answered the item. It ranges from 0 to 100%. The higher the percentage, the easier the item. The recommended range of difficulty is from 30 to 70% [44]. The mean item discrimination index of the items was 0.34 and the item discrimination index of most items was greater than 0.33, implying that the items had discrimination validity [45]. Discrimination index describes the ability of an item to distinguish between high and low scorers. The value of the discrimination index can range from  $-1.00$  to  $+1.00$ . A highly discriminating item indicates that the students who had high tests scores got the item correct whereas students who had low test scores got the item incorrect. Items having negative discrimination are rejected [46].

The learning motivation questionnaire is amended from the intrinsic value scale of Motivated Strategies for Learning Questionnaire (MSLQ) [47,48]. It has nine questions and uses a seven-point Likert scale, which ranges from 1 (totally disagree) to 7 (totally agree), for instance, "I prefer class work that is challenging so I can learn new things", "It is important for me to learn what is being taught in this class", "I like what I am learning in this class", "I think I will be able to use what I learn in this class in other classes", "I often choose paper topics I will learn something from even if they require more work", "Even when I do poorly on a test I try to learn from my mistakes", "I think that what I am learning in this class is useful for me to know", "I think that what we are learning in this class is interesting", and "Understanding this subject is important to me". The overall Cronbach's alpha value of the questionnaire was 0.92. The Cronbach's alpha reliability coefficient is

used to calculate the internal consistency of an attitude scale or an achievement test when a test measures a unidimensional trait and if it is scored by ratings [43].

The learning attitude questionnaire was developed by Hwang and Chang [49]. The questionnaire consisted of seven items scored using a four-point rating scale, which ranges from 1 (strongly disagree) to 4 (strongly agree), for instance, “The course is valuable and worth studying” and “I would like to know more about the learning targets”. The overall Cronbach’s alpha value of the questionnaire was 0.87.

The problem solving ability questionnaire was developed to evaluate the problem solving ability of students while developing software, and was employed to compare the problem solving ability of students in the three different software engineering learning approaches [18]. It consists of 25 items with five-point Likert rating scale ranging from 1 (totally disagree) to 5 (totally agree), for instance, “When I encounter a problem, I will first explore the key to the problem”, “When I encounter problems, I will think about what to do next”, and “I can often come up with innovative and effective ways to solve problems”. The overall Cronbach’s alpha value of the questionnaire was 0.75.

#### 4.3. Procedure

The overall procedure of this study is described in Figure 5. The experiment of each class had a length of 10 weeks (21 h). Before the experiment, the students in the experimental group 1, experimental group 2, and control group were asked to take four pre-tests. The first pre-test was a learning motivation questionnaire and the second was a learning attitude questionnaire. Thirdly, all students were then asked to fill out a problem solving ability questionnaire. Following that, the students completed a pre-test to evaluate the equivalent of the students’ prior knowledge of software engineering.

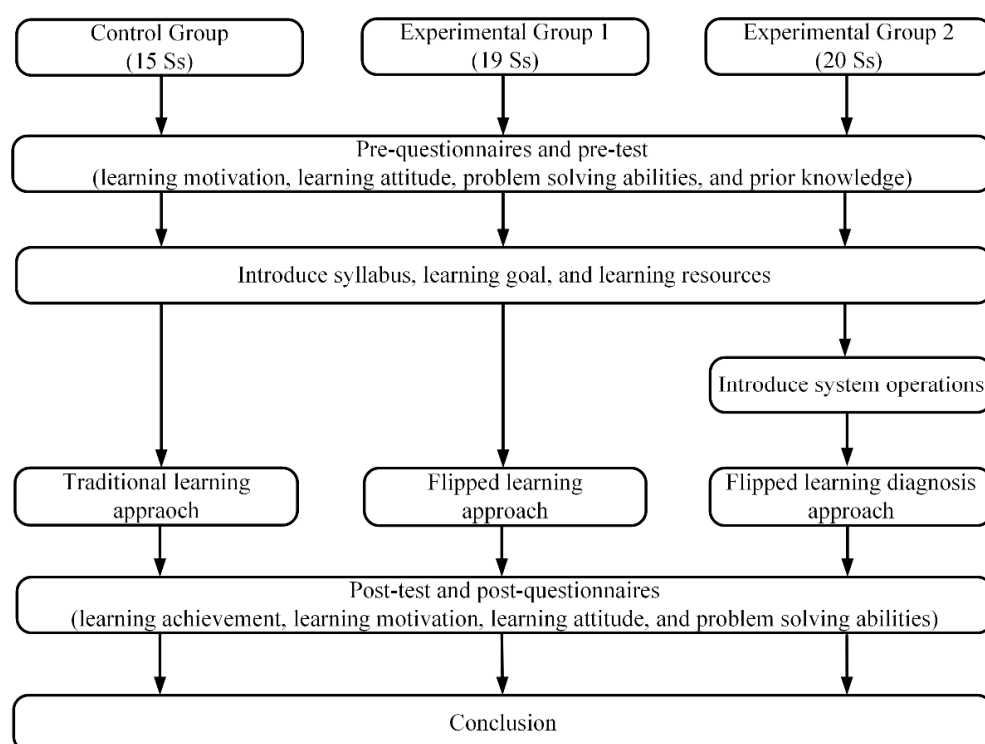


Figure 5. The experimental process.

Before undergoing the formal learning process, the teacher firstly took 30 min to explain the traditional teaching approach to the control group and explain the flipped learning approach to the two experimental groups. Moreover, the teacher additionally took 20 min to introduce the operations of the proposed system to the experimental group 2.

During the process of the activities, the students in the control group received theoretical concepts of software engineering from the teacher by using slides in the traditional classroom. Meanwhile, the teacher assigned case studies, discussions, and practices to conduct high-order thinking activities using the remaining time in class. Moreover, in every week, the teacher assigned two open-ended questions (e.g., “Please explain what is different between agile method and traditional method with regard to software development and deploy”) related to the course and asked the students of the control group to complete the questions out of class.

On the contrary, the students from the experimental groups 1 and 2 learned with the flipped learning approach. Out of class, the students watched 61 video clips to learn fundamental knowledge of software engineering. Table 1 shows the information of the video clips with regard to the subject-matter area of software engineering in the experiment. The 61 videos were published on YouTube as the web link: <https://www.youtube.com/playlist?list=PLmFShmJuCYrvKlrTCDuQx6BzGHELNkJNh> (accessed on 31 August 2021). Moreover, the students of the experimental group 2 could use the proposed system to take specific diagnosis assessments to diagnose their learning problems and promote learning performance out of class. In class, the teacher guided the students in both experimental groups to engage in case studies, discussions, and practice activities. As well, the teacher also assigned two open-ended questions related to the course to the students of the two groups each week.

**Table 1.** The information on the video clips in the software engineering course.

#	Unit	Duration	#	Unit	Duration
01	Introduction	07:42	32	Structural Models Part 2	04:49
02	Software Processes	04:59	33	Behavioral Models Part 1	04:46
03	Software Process Models Part 1	07:24	34	Behavioral Models Part 2	05:47
04	Software Process Models Part 2	06:16	35	Design and Implementation	02:11
05	Software Process Models Part 3	05:32	36	Object-Oriented Design using the UML Part 1	02:57
06	Process Activities—Software Specification	05:24	37	Object-Oriented Design using the UML Part 2	04:46
07	Process Activities—Software Design and Implementation	03:07	38	Object-Oriented Design using the UML Part 3	03:05
08	Process Activities—Software Validation	03:44	39	Object-Oriented Design using the UML Part 4	02:45
09	Process Activities—Software Evolution	01:34	40	Object-Oriented Design using the UML Part 5	06:38
10	Coping with Change	02:35	41	Object-Oriented Design using the UML Part 6	02:54
11	Prototyping	02:04	42	Implementation Issues	08:43
12	Agile Software Development Part 1	05:03	43	Software Testing Part 1	05:11
13	Agile Software Development Part 2	06:19	44	Software Testing Part 2	05:25
14	Agile Software Development Part 3	06:01	45	Testing Processes	02:47
15	Extreme Programming Part 1	06:46	46	Development Testing Part 1	02:28
16	Extreme Programming Part 2	06:48	47	Development Testing Part 2	03:53
17	Agile Project Management	02:29	48	Development Testing Part 3	04:46
18	Scrum Method	06:09	49	Development Testing Part 4	02:44
19	Requirements Engineering	03:08	50	Development Testing Part 5	06:15
20	Functional and Non-Functional Requirements	06:26	51	Development Testing Part 6	04:20
21	The Software Requirements Document	03:25	52	Release Testing Part 1	03:26
22	Requirements Specification Part 1	05:52	53	Release Testing Part 2	04:52
23	Requirements Specification Part 2	03:14	54	Software Evolution	04:08
24	Requirements Elicitation and Analysis Part 1	07:51	55	Evolution Processes Part 1	03:43
25	Requirements Elicitation and Analysis Part 2	05:22	56	Evolution Processes Part 2	05:38
26	Requirements Validation	04:50	57	Software Maintenance Part 1	09:14
27	Requirements Management	04:57	58	Software Maintenance Part 2	04:15
28	System Modeling	05:02	59	Software Maintenance Part 3	04:34
29	Context Models	01:32	60	Software Maintenance Part 4	04:17
30	Interaction Models	07:44	61	Software Maintenance Part 5	08:47
31	Structural Models Part 1	06:05			



After the experiment and learning activities, all three groups received three questionnaires and a post-test to measure their learning motivation, learning attitude, problem solving ability, and learning achievement.

#### 4.4. Research Hypotheses

- (1) The flipped learning diagnosis approach is not more effective than the traditional learning approach in terms of student learning achievement.
- (2) The flipped learning approach is not more effective than the traditional learning approach in terms of student learning achievement.
- (3) The flipped learning diagnosis approach is not more effective than the traditional learning approach in terms of student learning motivation.
- (4) The flipped learning approach is not more effective than the traditional learning approach in terms of student learning motivation.
- (5) The flipped learning diagnosis approach is not more effective than the traditional learning approach in terms of student learning attitude.
- (6) The flipped learning approach is not more effective than the traditional learning approach in terms of student learning attitude.
- (7) The flipped learning diagnosis approach is not more effective than the traditional learning approach in terms of student problem solving ability.
- (8) The flipped learning approach is not more effective than the traditional learning approach in terms of student problem solving ability.

#### 4.5. Data Analysis

IBM SPSS was used to analyze the performance of the students in the experiment, including the results of the learning achievement, learning motivation, learning attitude, problem solving ability, and learning behaviors. This study applied the analysis of variance (ANOVA) and analysis of covariance (ANCOVA) models to compare the mean response between the three groups in terms of learning achievement, learning motivation, learning attitude, and problem solving ability quantitative data. Moreover, the paired sample *t*-test was used to examine the differences in the learning motivation, learning attitude and problem solving ability for the three groups before and after the learning process. Linear regression was applied to analyze the relations between students' learning behaviors and learning achievements.

#### 4.6. Research Ethics

This study was carried out in accordance with the ethical principles of the Ministry of Science and Technology, Taiwan, R.O.C. and the Human Research Ethics Committee of National Cheng Kung University.

### 5. Results

#### 5.1. Learning Achievement

This study first investigated three learning approaches, the traditional learning approach, the flipped learning approach, and the flipped diagnosis learning approach, to find out whether there are significant differences in the learning achievements when students study software engineering courses. In terms of prior knowledge, the one-way ANOVA was applied to analyze the pre-test results between the three groups. The result revealed that there was no significant difference ( $F(1, 51) = 2.704, p = 0.077 > 0.05$ ) between the pre-test scores of the three groups, indicating that they had identical software engineering learning performance before the experiment.

This study further used the one-way ANCOVA to do the analysis, with the pre-test as the covariance, the learning approaches as the independent variables, and the post-test as the dependent variables. The homogeneity of the regression slopes was confirmed, indicating that it was appropriate to apply the analysis of covariance ( $F = 1.871, p = 0.165 > 0.05$ ). The results of the ANCOVA analysis are shown in Table 2. According to Table 2, the adjusted

means and standard deviation were 77.20 and 2.89 for the experimental group 1, 83.99 and 2.79 for the experimental group 2, and 70.88 and 3.33 for the control group. It was found that there was a statistically significant difference between the adjusted means ( $F(1, 50) = 4.552$ ,  $p = 0.015 < 0.05$ ). The post-hoc analysis was conducted, and it was found that there was a significant difference between the experimental group 2 and the control group. The result reveals that the flipped learning approach with the proposed system benefited the students more than the traditional teaching approach with regard to the learning achievement. Therefore, hypothesis 1 was rejected and hypothesis 2 was accepted.

**Table 2.** The ANCOVA results for the students' learning achievements.

Group	N	M	SD	Adjusted M	Adjusted SD	F	Pairwise Comparisons
(1) Experimental Group 1	19	76.31	14.22	77.20	2.89	4.552	(2) > (3)
(2) Experimental Group 2	20	83.50	9.88	83.99	2.79		
(3) Control Group	15	72.66	14.37	70.88	3.33		

### 5.2. Learning Motivation

In order to explore the students' performance of learning motivation, the one-way ANCOVA was adopted. A Shapiro–Wilk test was used to test determine whether the data collected was normally distributed or not. Due to  $p$ -values for all groups being larger than 0.05, it means the data are normally distributed [50]. The skewness and kurtosis from all scores obtained from the groups were calculated. The variation of skewness values varied from  $-1.01$  to  $0.21$  and the variation of kurtosis values varied from  $-1.60$  and  $0.52$ . According to Kline (2005), the limit values of skewness and kurtosis should not be more than 3.0 and 10.0, respectively. It can be adduced from the obtained values that the hypothesis of normality was satisfied in this sense [51]. The homogeneity of the regression slopes was confirmed, indicating that it was appropriate to employ the analysis of covariance ( $F = 0.126$ ,  $p = 0.882 > 0.05$ ). The result of analyzing the students' learning motivation is shown in Table 3. According to the result, there is a significant difference in the students' learning motivation among the three groups ( $F(1, 50) = 4.027$ ,  $p = 0.024 < 0.05$ ). The post-hoc analysis was conducted, and it was found that there was a significant difference between the experimental group 2 and the control group. The result implies that the flipped learning approach with the proposed system significantly benefited the students more than the traditional teaching approach in terms of the learning motivation. Therefore, hypothesis 3 was rejected and hypothesis 4 was accepted.

**Table 3.** The ANCOVA results for the students' learning motivation.

Group	N	M	SD	Adjusted M	Adjusted SD	F	Pairwise Comparisons
(1) Experimental Group 1	19	5.71	0.65	5.84	0.12	4.027	(2) > (3)
(2) Experimental Group 2	20	5.99	0.76	5.88	0.11		
(3) Control Group	15	5.41	0.62	5.41	0.13		

To examine the difference in the learning motivation for the three groups before and after the learning process, the paired sample  $t$ -test was used to analyze them. Table 4 shows that the students in the experimental group 1 were motivated after participating in the learning activities. The students in the experimental group 2 and the control group did not show the significant difference before and after participating in the learning process with regard to the learning motivation. However, it is noteworthy that the learning motivation of the students in the control group had slightly decreased after participating in the course.

**Table 4.** The paired sample *t*-test results of learning motivation for the three groups of students.

Group	Tests	N	M	SD	<i>t</i>	<i>p</i>
(1) Experimental Group 1	Pre-test	19	5.31	0.73	−2.817	0.011 *
	Post-test	19	5.71	0.65		
(2) Experimental Group 2	Pre-test	20	5.73	0.83	−1.827	0.084
	Post-test	20	5.99	0.76		
(3) Control Group	Pre-test	15	5.54	0.82	0.087	0.433
	Post-test	15	5.41	0.62		

\*  $p < 0.05$ .

### 5.3. Learning Attitude

In order to evaluate the effects of the different learning approaches on their learning attitude, the one-way ANCOVA was also conducted. The homogeneity of the regression slopes was not violated ( $F = 0.215$ ,  $p = 0.807 > 0.05$ ). The ANCOVA result of analyzing the students' learning attitude is shown in Table 5. According to the result, there is a significant difference in the students' learning attitude among the three groups ( $F(1, 50) = 6.708$ ,  $p = 0.003 < 0.05$ ). The post-hoc analysis was conducted, and it was found that there was a significant difference between the experimental group 1, the experimental group 2, and the control group. The result reveals that the flipped learning approach benefited the students more than the traditional teaching approach with regard to the learning attitude. Therefore, hypothesis 5 was rejected and hypothesis 6 was rejected.

**Table 5.** The ANCOVA results for the students' learning attitudes.

Group	N	M	SD	Adjusted M	Adjusted SD	<i>F</i>	Pairwise Comparisons
(1) Experimental Group 1	19	3.33	0.41	3.50	0.08	6.708	(1), (2) > (3)
(2) Experimental Group 2	20	3.48	0.41	3.43	0.07		
(3) Control Group	15	3.24	0.34	3.08	0.08		

Furthermore, to examine the difference in the learning attitude for the three groups before and after the learning process, the paired sample *t*-test was used to analyze them. Table 6 shows that the students in the experimental group 1 had a significant positive effect on their learning attitude after participating in the learning activities. However, the students in the control group showed a significant negative effect on their learning attitude after participating in the learning activities.

**Table 6.** The paired sample *t*-test results of learning attitude for the three groups of students.

Group	Tests	N	M	SD	<i>t</i>	<i>p</i>
(1) Experimental Group 1	Pre-test	19	3.04	0.316	−3.884 *	0.001
	Post-test	19	3.33	0.412		
(2) Experimental Group 2	Pre-test	20	3.36	0.391	−1.538	0.141
	Post-test	20	3.48	0.418		
(3) Control Group	Pre-test	15	3.51	0.250	3.392 *	0.004
	Post-test	15	3.24	0.340		

\*  $p < 0.05$ .

### 5.4. Problem Solving Ability

To examine how the different learning approaches influenced the students' problem solving ability in this study, the one-way ANCOVA was adopted. The homogeneity of the regression slopes was confirmed, indicating that it was appropriate to employ the analysis of covariance ( $F = 2.539$ ,  $p = 0.090 > 0.05$ ). The results of the ANCOVA analysis are shown in Table 7. Based on the analysis results, there is a statistically significant difference between adjusted means ( $F(1, 50) = 3.964$ ,  $p = 0.025 < 0.05$ ). The post-hoc analysis was conducted,

and it was found that there was a significant difference between the experimental group 2 and the control group. The result implies that the flipped learning approach with the proposed system significantly benefited the students more than the traditional teaching approach in terms of the problem solving ability. Therefore, hypothesis 7 was rejected and hypothesis 8 was accepted.

**Table 7.** The ANCOVA results for the students' problem solving abilities.

Group	N	M	SD	Adjusted M	Adjusted SD	F	Pairwise Comparisons
(1) Experimental Group 1	19	3.73	0.31	3.50	0.07	3.964	(2) > (3)
(2) Experimental Group 2	20	3.81	0.40	3.76	0.07		
(3) Control Group	15	3.50	0.18	3.77	0.06		

To examine the difference in the problem solving ability for the three groups before and after the learning process, the paired sample *t*-test was used to analyze them. Table 8 shows that the students in the experimental group 2 experienced a significant positive effect on their problem solving ability after participating in the course. The students in the experimental group 1 and the control group did not show the significant difference before and after participating in the learning process with regard to the problem solving ability. However, the learning problem solving ability of the students in the control group had slightly decreased after participating in the course.

**Table 8.** The paired *t*-test results of problem solving ability for the three groups of students.

Group	Tests	N	M	SD	<i>t</i>	<i>p</i>
(1) Experimental Group 1	Pre-test	19	3.45	0.178	−1.484	0.155
	Post-test	19	3.58	0.361		
(2) Experimental Group 2	Pre-test	20	3.62	0.289	−3.057 *	0.006
	Post-test	20	3.83	0.375		
(3) Control Group	Pre-test	15	3.52	0.328	0.221	0.828
	Post-test	15	3.50	0.180		

\*  $p < 0.05$ .

### 5.5. Analysis of Learning Behaviors and Learning Achievement

The experimental results revealed that the flipped learning approach with the proposed system can benefit teachers and students to teach and learn software engineering. Regarding the understanding of the impacts on students' flipped learning when using the proposed system, this study further investigated students' learning behaviors by using learning logs on the proposed system. The learning logs involved the time to watch videos, the frequency of diagnostic assessments, and the number of understanding concepts. The time to watch videos means that a student accumulates watch time (seconds) of the total videos during the entire learning process. The frequency of diagnostic assessments means the frequency of diagnostic assessments taken by a student during the entire learning process. In the conducted course, the teacher developed 10 diagnostic assessments and the students could repeatedly take the assessments during the course session. The number of understanding concepts means the number of concepts understood by a student. The teacher planned 30 concepts to facilitate the students to learn the concepts in the course.

To examine the relations between each learning behavior and learning achievement, the linear regression approach was applied. The regression coefficients and standard errors of the relation between each learning behavior and learning achievement are presented in Table 9. The results of the linear regression approach show that the time to watch videos and the number of understanding concepts tested were statistically significant predictors of the learning achievement. It is noteworthy that the frequency of diagnostic assessments does not significantly predict student learning achievement.

**Table 9.** The linear regression results for the relations between each learning behavior and learning achievement.

Factor	R <sup>2</sup>	Adjusted R <sup>2</sup>	B	t	p
Time to watch videos	0.673	0.655	0.001	6.087 *	0.000
Frequency of diagnostic assessments	0.189	0.144	0.986	2.049	0.055
Number of understanding concepts	0.530	0.504	1.295	4.503*	0.000

\*  $p < 0.05$ .

## 6. Discussion and Conclusions

Traditional software engineering education makes scant provision for assisting students in keeping their software engineering knowledge current. In this study, the flipped learning strategy with the flipped learning diagnosis system was proposed to promote software engineering education. Flipped learning has been regarded as a learning strategy that encourages high-order thinking activities in the class. Many studies have confirmed its effectiveness for students' learning performances; in addition, researchers have been interested in providing suitable learning services to support students' learning out of class in the flipped classroom [5,52,53]. To explore students' learning performance in software engineering courses, in this study three classes of students were invited to learn with three different software engineering learning approaches (traditional learning approach, flipped learning approach, and flipped learning diagnosis approach), and their learning achievement, learning motivation, learning attitude, problem solving ability, and learning behaviors were further investigated.

According to the results of the students' learning achievements, it was found that most of the students who learned with the flipped learning diagnosis approach outperformed those students who learned with the traditional learning approach. This result responded to research question 1. This result is also consistent with the research findings of Chang and Hwang [26], who found that the students who engaged in appropriate diagnosis activities could improve their learning achievement in the flipped classroom. In the meantime, this study found that the students who learned with the flipped learning diagnosis approach had higher problem solving ability than those students who learned with the traditional learning approach. This result responded to research question 3. This result also supports the argument proposed by Wang [54], who indicated that self-assessment and self-reflection activities promote students' problem solving ability in the flipped classroom. The result of the students' learning performances also showed that in addition to problem solving ability, most of the students in the flipped learning group had similar learning motivation and learning attitude to the students in the flipped learning diagnosis group. This result responded to research question 2. Many studies have confirmed the effectiveness of the flipped learning strategy for students' learning motivation and learning attitude [5,55,56]. In addition, most of the students in the flipped learning group learned better than the students in the traditional learning group. On the other hand, the traditional learning group students did not perform well in comparison with the other two flipped learning groups, especially for those students who had negative effects on learning motivation, learning attitude, and problem solving ability. This supports the argument proposed by Eccles et al. [57], who indicated that the traditional learning approach had negative effects on students' learning motivation due to teachers generally providing fewer interaction and discussion opportunities in the teacher-centered learning environment. To further use the linear regression approach to examine the relation between learning motivation and learning attitude, the result shows that the learning motivation tested was statistically a significant predictor of the learning attitude ( $R^2 = 0.278$ , Adjusted  $R^2 = 0.264$ ,  $B = 0.296$ ,  $t = 4.470$ ,  $p = 0.000 < 0.05$ ). This implies that the students' learning attitude was positively affected by the students' learning motivation. In addition, the learning logs derived from the students in the flipped learning diagnosis group had two significant factors (time to watch videos and number of understanding concepts) to predict students' learning achievement. This result responded to research question 4. It



is interesting that the frequency of diagnostic assessments did not significantly predict student learning achievement. The reason inferred from this study is that some of the students may not take the assessments seriously each time.

In sum, one contribution of the present study is to demonstrate an effective way of conducting the flipped learning strategy in software engineering courses from both the perspectives of learning performances and behaviors. Another contribution of the present study is to show that the flipped learning diagnosis approach can encourage students to engage in the learning process well, and hence enable them to have better learning performance after participating in the course. Accordingly, it is suggested that, for those teachers who intend to apply the flipped learning strategy to software engineering courses, it is important to consider using suitable learning tools or services out of class in the learning design. Moreover, the literature indicates that it is preferable to implement the flipped learning approach in a small class (<20 students) since it is possible to involve all the students in class activities at one time [58,59].

On the other hand, this study has some limitations that should be noted. First, as a result of ethical and practical considerations, a random selection of participants was not used for the study. Students were assigned into the control and experimental groups based on the class session they were enrolled in. Second, the teacher who taught the three classes might have had preconceived ideas about the different treatments, which might have affected the students' learning outcomes. Therefore, the estimation of the effect of the flipped learning was subject to possible contamination by confounding variables. Third, this study evaluated the group performances rather than the individual performances. Finally, the lack of generalizability is also a limitation of the present data as the sample size was not large. Future studies should implement random selections and conduct in-depth interviews when applicable. In addition, various samples should be increased in the future in collaboration with other universities and institutions.

**Funding:** This research was funded by Ministry of Science and Technology, Taiwan, R.O.C., grant numbers MOST 104-2511-S-153-002-MY2, MOST 106-2511-S-153-003-MY2, MOST 108-2511-H-153-006-MY2, MOST 108-2745-8-153-001, MOST 109-2511-H-153-007, and MOST 110-2511-H-153-002-MY3.

**Institutional Review Board Statement:** The study was conducted according to the guidelines of the Declaration of Ministry of Science and Technology, Taiwan, R.O.C., and approved by the Human Research Ethics Committee of National Cheng Kung University (protocol code 108-289 and 9 November 2019 of approval).

**Informed Consent Statement:** Informed consent was obtained from all subjects involved in the study.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The author declares no conflict of interest.

## References

1. Ford, G.A.; Gibbs, N.E. A master of software engineering curriculum: Recommendations from the software engineering institute. *Computer* **1989**, *22*, 59–71. [\[CrossRef\]](#)
2. Cico, O.; Jaccheri, L.; Nguyen-Duc, A.; Zhang, H. Exploring the intersection between software industry and software engineering education—A systematic mapping of software engineering trends. *J. Syst. Softw.* **2021**, *172*, 110736. [\[CrossRef\]](#)
3. Ardis, M.; Budgen, D.; Hislop, G.W.; Offutt, J.; Sebern, M.; Visser, W. SE 2014: Curriculum guidelines for undergraduate degree programs in software engineering. *Computer* **2015**, *48*, 106–109. [\[CrossRef\]](#)
4. Mauricio, R.d.A.; Veadó, L.; Moreira, R.T.; Figueiredo, E.; Costa, H. A systematic mapping study on game-related methods for software engineering education. *Inf. Softw. Technol.* **2018**, *95*, 201–218.
5. Lin, Y.N.; Hsia, L.H.; Hwang, G.J. Promoting pre-class guidance and in-class reflection: A SQIRC-based mobile flipped learning approach to promoting students' billiards skills, strategies, motivation and self-efficacy. *Comput. Educ.* **2021**, *160*, 104035.
6. McGuinness, N.; Vlachopoulos, D. Student experiences of using online material to support success in A-level economics. *Int. J. Technol. Learn.* **2019**, *14*, 80–109. [\[CrossRef\]](#)

7. Zhao, L.; Liu, X.; Su, Y.-S. The Differentiate effect of self-efficacy, motivation, and satisfaction on pre-service teacher students' learning achievement in a flipped classroom: A case of a modern educational technology course. *Sustainability* **2021**, *13*, 2888. [CrossRef]
8. Brewer, R.; Movahedazarhouli, S. Flipped learning in flipped classrooms: A new pathway to prepare future special educators. *J. Digit. Learn. Teach. Educ.* **2019**, *35*, 128–143. [CrossRef]
9. Akçayır, G.; Akçayır, M. The flipped classroom: A review of its advantages and challenges. *Comput. Educ.* **2018**, *126*, 334–345. [CrossRef]
10. Jensen, J.L.; Holt, E.A.; Sowards, J.B.; Ogden, T.H.; West, R.E. Investigating strategies for pre-class content learning in a flipped classroom. *J. Sci. Educ. Technol.* **2018**, *27*, 523–535. [CrossRef]
11. Shahzad, B.; Fazal-e-Amin; Abro, A.; Imran, M.; Shoaib, M. Resource optimization-based software risk reduction model for large-scale application development. *Sustainability* **2021**, *13*, 2602. [CrossRef]
12. Shahzad, B.; Javed, I.; Shaikh, A.; Sulaiman, A.; Abro, A.; Ali Memon, M. Reliable requirements engineering practices for COVID-19 using blockchain. *Sustainability* **2021**, *13*, 6748. [CrossRef]
13. Dingsøyr, T.; Moe, N.B.; Fægri, T.E.; Seim, E.A. Exploring software development at the very large-scale: A revelatory case study and research agenda for agile method adaptation. *Empir. Softw. Eng.* **2018**, *23*, 490–520. [CrossRef]
14. Lin, Y.T. When mobile technology meets traditional classroom learning environment: How does it improve students' learning performances? In *Learning Environments: Emerging Theories, Applications and Future Directions*; Wallace, K., Ed.; Nova Science Publishers Inc.: New York, NY, USA, 2016; Chapter 8.
15. Beecham, S.; Clear, T.; Noll, J. Do we teach the right thing?: A comparison of global software engineering education and practice. In Proceedings of the 2017 IEEE 12th International Conference on Global Software Engineering (ICGSE), Buenos Aires, Argentina, 22–23 May 2017; pp. 11–20.
16. Alhammad, M.M.; Moreno, A.M. Gamification in software engineering education: A systematic mapping. *J. Syst. Softw.* **2018**, *141*, 131–150. [CrossRef]
17. Mishra, A.; Cagiltay, N.E.; Kilic, O. Software engineering education: Some important dimensions. *Eur. J. Eng. Educ.* **2007**, *32*, 349–361. [CrossRef]
18. Lin, Y.T. Impacts of a flipped classroom with a smart learning diagnosis system on students' learning performance, perception, and problem solving ability in a software engineering course. *Comput. Hum. Behav.* **2019**, *95*, 187–196. [CrossRef]
19. Malik, B.; Zafar, S. A systematic mapping study on software engineering education. *Int. J. Sci. Res. Innov.* **2012**, *6*, 3343–3353.
20. Garousi, V.; Petersen, K.; Ozkan, B. Challenges and best practices in industry-academia collaborations in software engineering: A systematic literature review. *Inf. Softw. Technol.* **2016**, *79*, 106–127. [CrossRef]
21. Lin, Y.T.; Lin, Y.C. Applying mobile technology to the support learning and diagnosis approach in a flipped classroom. In *Mobile Learning: Students' Perspectives, Applications and Challenges*; René, D., Aubin, C., Eds.; Nova Science Publishers Inc.: New York, NY, USA, 2017; Chapter 5.
22. Bull, C.N.; Whittle, J. Supporting reflective practice in software engineering education through a studio-based approach. *IEEE Softw.* **2014**, *31*, 44–50. [CrossRef]
23. López-Pimentel, J.C.; Medina-Santiago, A.; Alcaraz-Rivera, M.; Del-Valle-Soto, C. Sustainable project-based learning methodology adaptable to technological advances for web programming. *Sustainability* **2021**, *13*, 8482. [CrossRef]
24. Pappas, I.O.; Mora, S.; Jaccheri, L.; Mikalef, P. Empowering social innovators through collaborative and experiential learning. In Proceedings of the 2018 IEEE Global Engineering Education Conference (EDUCON), Santa Cruz de Tenerife, Spain, 17–20 April 2018; pp. 1080–1088.
25. Wong, K. Experiences in constructing a MOOC specialization. In Proceedings of the 21st Western Canadian Conference on Computing Education, Kamloops, BC, Canada, 6–7 May 2016; ACM: New York, NY, USA, 2016; Volume 19, pp. 1–4.
26. Chang, S.C.; Hwang, G.J. Impacts of an augmented reality-based flipped learning guiding approach on students' scientific project performance and perceptions. *Comput. Educ.* **2018**, *125*, 226–239. [CrossRef]
27. Khan, S. Let's Use Video to Reinvent Education. 2011. Available online: [http://www.ted.com/talks/salman\\_khan\\_let\\_s\\_use\\_video\\_to\\_reinvent\\_education](http://www.ted.com/talks/salman_khan_let_s_use_video_to_reinvent_education) (accessed on 31 August 2021).
28. Collado-Valero, J.; Rodríguez-Infante, G.; Romero-González, M.; Gamboa-Ternero, S.; Navarro-Soria, I.; Lavigne-Cerván, R. Flipped classroom: Active methodology for sustainable learning in higher education during social distancing due to COVID-19. *Sustainability* **2021**, *13*, 5336. [CrossRef]
29. Bloom, B.; Englehart, M.; Furst, E.; Hill, W.; Krathwohl, D. *Taxonomy of Educational Objectives: The Classification of Educational Goals, Handbook I: Cognitive Domain*; Longmans, Green: New York, NY, USA, 1956.
30. Zheng, L.; Bhagat, K.K.; Zhen, Y.; Zhang, X. The effectiveness of the flipped classroom on students' learning achievement and learning motivation: A meta-analysis. *J. Educ. Technol. Soc.* **2020**, *23*, 1–15.
31. Pozo-Sánchez, S.; López-Belmonte, J.; Fuentes-Cabrera, A.; López-Núñez, J.-A. Twitch as a techno-pedagogical resource to complement the flipped learning methodology in a time of academic uncertainty. *Sustainability* **2021**, *13*, 4901. [CrossRef]
32. Alias, M.; Iksan, Z.; Karim, A.; Nawawi, A.; Nawawi, S. A novel approach in problem-solving skills using flipped classroom technique. *Creat. Educ.* **2020**, *11*, 38–53. [CrossRef]
33. Liang, H.-Y.; Hsu, T.-Y.; Hwang, G.-J.; Chang, S.-C.; Chu, H.-C. A mandatory contribution-based collaborative gaming approach to enhancing students' collaborative learning outcomes in Science museums. *Interact. Learn. Environ.* **2021**, in press. [CrossRef]

34. Evseeva, A.; Solozhenko, A. Use of flipped classroom technology in language learning. *Procedia-Soc. Behav. Sci.* **2015**, *206*, 205–209. [\[CrossRef\]](#)
35. Al-Harbi, S.; Alshumaimeri, Y. The flipped classroom impact in grammar class on EFL Saudi Secondary School students' performances and attitudes. *Engl. Lang. Teach.* **2016**, *9*, 60–80. [\[CrossRef\]](#)
36. Hsieh, J.S.C.; Wu, W.-C.V.; Marek, M.W. Using the flipped classroom to enhance EFL learning. *Comput. Assist. Lang. Learn.* **2017**, *30*, 1–21. [\[CrossRef\]](#)
37. Ho, J. Gamifying the flipped classroom: How to motivate Chinese ESL learners? *Innov. Lang. Learn. Teach.* **2020**, *14*, 421–435. [\[CrossRef\]](#)
38. Lo, C.K.; Hew, K.F. A critical review of flipped classroom challenges in K-12 education: Possible solutions and recommendations for future research. *Res. Pract. Technol. Enhanc. Learn.* **2017**, *12*, 4. [\[CrossRef\]](#)
39. Chang, Y.-H.; Lin, J.-Y.; Lu, Y.-T. Enhancing the intention to preview learning materials and participate in class in the flipped classroom context through the use of handouts and incentivisation with virtual currency. *Sustainability* **2021**, *13*, 3276. [\[CrossRef\]](#)
40. Lin, Y.C.; Lin, Y.T.; Huang, Y.M. Development of a diagnostic system using a testing-based approach for strengthening student prior knowledge. *Comput. Educ.* **2011**, *57*, 1557–1570. [\[CrossRef\]](#)
41. Lin, Y.C.; Huang, Y.M. A fuzzy-based prior knowledge diagnostic model with multiple attribute evaluation. *J. Educ. Technol. Soc.* **2013**, *16*, 119–136.
42. Wei, C.W.; Lin, Y.C.; Lin, Y.T. An interactive diagnosis approach for supporting clinical nursing courses. *Interact. Learn. Environ.* **2016**, *24*, 1795–1811. [\[CrossRef\]](#)
43. Tan, S. Misuses of KR-20 and Cronbach's alpha reliability coefficients. *Egit. Bilim* **2009**, *34*, 101–112.
44. Hingorjo, M.R.; Jaleel, F. Analysis of one-best MCQs: The difficulty index, discrimination index and distractor efficiency. *J. Pak. Med. Assoc.* **2012**, *62*, 142–147.
45. Doran, R. *Basic Measurement and Evaluation of Science Instruction*; National Science Teachers Association: Washington, DC, USA; New York, NY, USA, 1980.
46. Boopathiraj, C.; Chellamani, K. Analysis if the test items on difficulty level and discrimination coefficient in the test for research in education. *IJSSIR* **2013**, *2*, 189–193.
47. Lin, Y.T.; Tseng, Y.M.; Lee, Y.S.; Wang, T.C.; Tsai, S.I.; Yi, Y.J. Development of a SoLoMo game-based application for supporting local cultural learning in Taiwan. *J. Educ. Technol. Soc.* **2018**, *21*, 115–128.
48. Pintrich, P.R.; De Groot, E.V. Motivational and self-regulated learning components of classroom academic performance. *J. Educ. Psychol.* **1990**, *82*, 33–40. [\[CrossRef\]](#)
49. Hwang, G.J.; Chang, H.F. A Formative assessment-based mobile learning approach to improving the learning attitudes and achievements of students. *Comput. Educ.* **2011**, *56*, 1023–1031. [\[CrossRef\]](#)
50. Zhonggen, Y. Differences in serious game-aided and traditional English vocabulary acquisition. *Comput. Educ.* **2018**, *127*, 214–232. [\[CrossRef\]](#)
51. Kline, R.B. *Principles and Practice of Structural Equation Modeling*, 2nd ed.; Guilford Press: New York, NY, USA, 2005.
52. Chao, C.Y.; Chen, Y.T.; Chuang, K.Y. Exploring students' learning attitude and achievement in flipped learning supported computer aided design curriculum: A study in high school engineering education. *Comput. Appl. Eng. Educ.* **2015**, *23*, 514–526. [\[CrossRef\]](#)
53. Chiang, T.H.C.; Yang, S.J.H.; Yin, C. Effect of gender differences on 3-on-3 basketball games taught in a mobile flipped classroom. *Interact. Learn. Environ.* **2019**, *27*, 1093–1105. [\[CrossRef\]](#)
54. Wang, F.H. An exploration of online behaviour engagement and achievement in flipped classroom supported by learning management system. *Comput. Educ.* **2017**, *114*, 79–91. [\[CrossRef\]](#)
55. Bond, M. Facilitating student engagement through the flipped learning approach in K-12: A systematic review. *Comput. Educ.* **2020**, *151*, 103819. [\[CrossRef\]](#)
56. Lin, H.C.; Hwang, G.J.; Hsu, Y.D. Effects of ASQ-based flipped learning on nurse practitioner learners' nursing skills, learning achievement and learning perceptions. *Comput. Educ.* **2019**, *139*, 207–221. [\[CrossRef\]](#)
57. Eccles, J.; Wigfield, A.; Midgley, C.; Reuman, D.; Iver, D.M.; Feldlaufer, H. Negative effects of traditional middle schools on students' motivation. *Elem. Sch. J.* **1993**, *93*, 553–574. [\[CrossRef\]](#)
58. Galway, L.P.; Corbett, K.K.; Takaro, T.K.; Tairyan, K.; Frank, E. A novel integration of online and flipped classroom instructional models in public health higher education. *BMC Med. Educ.* **2014**, *14*, 181. [\[CrossRef\]](#)
59. Kerr, B. The flipped classroom in engineering education: A survey of the research. In Proceedings of the 2015 International Conference on Interactive Collaborative Learning Proceedings, Florence, Italy, 20–24 September 2015.