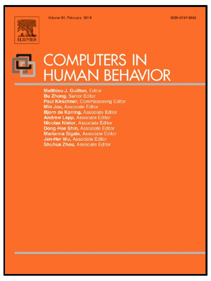

Impacts of a flipped classroom with a smart learning diagnosis system on students' learning performance, perception, and problem solving ability in a software engineering course

Yen-Ting Lin

PII:	S0747-5632(18)30571-5
DOI:	10.1016/j.chb.2018.11.036
Reference:	CHB 5807
To appear in:	*Computers in Human Behavior*
Received Date:	10 April 2018
Accepted Date:	17 November 2018

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

Yen-Ting Lin [*]

Department of Computer Science, National Pingtung University, No. 4-18 Minsheng Rd., Pingtung City, Pingtung County 90003, Taiwan (R.O.C.)

E-mail: ricky014@gmail.com

[*] Corresponding Author: Yen-Ting Lin
Email: ricky014@gmail.com
Tel: +886-8-7663800#33550
Fax: +886-8-7215034

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

Abstract

In recent years, many institutions have announced the significance of software development for countries, societies, and individuals. In developing software, various unpredictable problems are often encountered, especially in developing large-scale and complex software. To reduce the possibility of these problems occurring, it is important for students to apply software engineering technology to scientifically define the criteria, models, and procedures needed in the software development process. Therefore, it is important to cultivate students to learn software engineering concepts and technologies. However, since the course duration is limited by the semester, most teachers can only conduct a teacher-centered learning environment to teach theoretical concepts to students. Most students cannot achieve high-order thinking skills and apply software engineering technology to solve practical problems after learning in this environment. As mentioned above, the aim of this study is to apply an innovative pedagogy, called a flipped classroom, to conduct a learner-centered learning environment in a software engineering course. Moreover, a smart learning diagnosis system was developed to support this pedagogy in this course. An experiment was conducted on a software engineering course at a university in Taiwan to investigate the effectiveness of the proposed approach. The students in the experimental group learned with the flipped-classroom learning approach, while the students in the control group learned with the traditional-classroom learning approach. The experimental results show that, in comparison with the traditional-classroom learning approach, the proposed approach significantly improved the students' learning achievement, learning motivation, learning attitude, and problem solving ability.

Keywords: flipped classroom; learning diagnosis; software engineering; improving classroom teaching

1. Introduction

In recent years, many institutions have announced the significance of software development for countries, societies, and individuals (Jeannette, 2006; Stross, 2012). Moreover, experts and scholars have also consistently opined that software will appear anywhere in the future (Andreessen, 2011; Meyer, Fritz, Murphy, & Zimmermann, 2014). However, in developing software, various unpredictable problems are often encountered, especially in developing large-scale and complex software. To reduce the possibility of these problems occurring, it is important for students to apply software engineering technology to scientifically define the criteria, models, and procedures needed in the software development process. The above points of view show the importance of software engineering (Sommerville, 2010). Therefore, it is also important to cultivate students in the departments of computer science or engineering to learn software engineering concepts and technologies (Hadjerrouit, 2005).

In general, to conduct software engineering courses, the instruction of theoretical concepts and practical skills in a learner-centered learning environment is an ideal teaching strategy for students (Lin & Lin, 2017). In the learner-centered learning environment, the teacher arranges appropriate learning activities to promote student engagement in thinking and further enhance their cognitive levels and problem solving abilities (Baeten, Kyndt, Struyven, & Dochy, 2010; Schultz, Duffield, Rasmussen, & Wageman, 2014; Voogt & Roblin, 2012). Many studies have proposed various approaches to conducting a learner-centered learning environment in different courses (Jou, Lin, & Tsai, 2016; Kong, 2015; Lin, 2016; Lin, Wen, Jou, & Wu, 2014). However, since the course duration is limited by the semester, most teachers can only conduct a teacher-centered learning environment to teach theoretical concepts to students (Baker, Navarro, & Van Der Hoek, 2005). In the teacher-centered learning environment, the teacher transmits knowledge to students directly, and the students are the recipients. When the students engage in such passive learning processes, most students are unlikely to engage in more complex thinking processes or develop high-order thinking skills. Therefore, it is difficult to apply software engineering technology to solving practical problems after they complete their educations (Jonassen, 2000; Bransford, Sherwood, Vye, & Rieser, 1986; Heppner & Petersen, 1982). Table 1 shows a list of characteristics of learner-centered and teacher-centered learning environments.

Table 1

List of characteristics of learner-centered and teacher-centered learning environments.

Perspective	Learner-centered	Teacher-centered
Participation	Students actively participate in learning process	Students passively participate in learning process
Knowledge	Students actively construct knowledge	Students passively receive knowledge
Interaction	Students interact with teacher and peers in learning process	Students learn alone in learning process
Thinking skills	Students are engaged in applying higher-order thinking skills in learning process, such as analysis, evaluation, and creation.	Students are engaged in applying lower-order thinking skills in learning process, such as remembering, understanding, and application

As mentioned above, the aim of this study is to apply an innovative pedagogy, called a flipped classroom, to conduct a learner-centered learning environment in a software engineering course. Moreover, a smart learning diagnosis system was developed to support this pedagogy in this course. An experiment was conducted on a software engineering course at a university in Taiwan to investigate the effectiveness of the proposed approach.

The remainder of this paper is organized as follows. Section 2 provides reviews of the theoretical background of this study. Section 3 describes the system developed in this study. The experiment and results evaluation are shown in Sections 4 and 5, respectively. Finally, the conclusions, discussions, and suggestions for further research are presented in Section 6.

2. Theoretical Background

2.1. Software Engineering Education

A software engineering course involves a range issues from software specification and lifecycle to software structure and programming. The purpose of a software engineering course is to enable students to use scientific methods to well define the criteria, models, and procedures needed in the software development process and then efficiently develop software that meets the needs of users (Sommerville, 2010). In Taiwan's higher education, to train students to become software developers, departments of computer science or engineering have planned relevant courses in programming to develop students' software development skills. Most students are able to acquire programming skills after participating in the courses. However, to develop satisfactory and robust software, programming skills alone are insufficient for students because programming is only part of the software development process (Moreno, Sanchez-Segura, Medina-Dominguez, &amp; Carvajal, 2012). Therefore, students also need to learn software engineering knowledge and technology to address software development and software project management issues.

To conduct software engineering courses, several studies have indicated that the course design should involve appropriate learning activities to conduct a learner-centered learning environment to facilitate teacher-student and student-student interactions to enhance students' learning performance (Chen & Teng, 2011; Hainey, Connolly, Stansfield, & Boyle, 2011; Maratou, Chatzidaki, & Xenos, 2016). Hainey et al. (2011) developed a game-based learning application to teach students how to collect and analyze user requirements during the system development process. This application provided a learner-centered learning environment to engage students in learning software engineering concepts in a virtual world. The research results showed that the game-based learning approach can effectively help students learn relevant concepts in the course. Furthermore, in general, to conduct a software project, a project team is an essential element. It is important how team members collaboratively develop the project during the software development process. Therefore, Chen and Teng (2011) proposed a learning system to support teachers and students to conduct collaborative learning and project-based learning in software engineering courses. The investigation result indicated that the proposed system can promote students to invest in a collaborative learning process to develop software projects collaboratively. In addition, Maratou et al. (2016) presented a role-play game to assist students in learning the software project management issue of software engineering in a three-dimensional online multiuser virtual world. The research result revealed that the proposed system can improve students' learning experience and performance.

As mentioned above, several studies have noted that software engineering courses should focus on the interaction between teachers and students and the discussion of and reflection about practical cases (Chamillard & Braun, 2002; Hadjerrouit, 2005; Saiedian, 2002). Moreover, educators also indicated that software engineering courses supplemented with appropriate computer technology can enhance students' problem solving ability (Chen & Teng, 2011).

2.2. Flipped Classroom

In traditional classrooms, teachers usually use the class time to teach the course material, and students usually receive the instruction in the class. Moreover, to promote students' thinking, teachers may assign homework or practice exercises to each student out of the class. However, to complete the assignment, students often need to have discussions with teachers and classmates to promote thinking skills. In other words, students cannot obtain sufficient learning resources to foster high-order thinking skills in traditional classrooms (Lin & Hwang, 2018a).

To change the traditional learning patterns and teacher-centered learning modes, the concept of the flipped classroom originated with Bergmann and Sams in 2007 (Bergmann & Sams, 2012). The flipped classroom is a learner-centered pedagogy that reverses the in-class and out-of-class learning activities in traditional classrooms (Chen, Wang, Kinshuk, & Chen, 2014). In the flipped classroom, the in-class lecture is transformed to before-class learning through videos or other forms of media to free up more in-class time for opportunities to discuss the issues, practice, or apply knowledge (Bergmann & Sams, 2014). Therefore, the flipped classroom can increase the interaction between teachers and students in class, give teachers the opportunity to address the problems of individual students, and enable students to have more successful experiences in knowledge application (Lin & Hwang, 2018b). To date, the flipped classroom has been applied to various educational degrees and courses (Slomanson, 2014; Teo, Tan, Yan, Teo, & Yeo, 2014).

Another important activity of the flipped classroom is students' self-learning prior to class since learning performance affects how instructors and students interact with the learning materials in class (Kim, Kim, Khera, & Getman, 2014). In self-learning activities prior to class, students may be unable to efficiently and effectively realize the gaps between their learning results and course aims. In this situation, students may face a strong risk of engaging in in-class learning activities on faulty foundations since they lack sufficient levels of prior knowledge (Lin & Huang, 2013).

As mentioned above, to support an ideal software engineering education, this study aims to apply the flipped-classroom pedagogy to conduct a learner-centered learning environment in a software engineering course. Moreover, since students' self-learning performance prior to class is significant in affecting their prior knowledge while conducting high-order thinking activities in class, this study develops a smart learning diagnosis system to support the flipped classroom to assist students in learning and diagnosing the theoretical concepts of software engineering and assist instructors in managing the students' learning status. To evaluate the effectiveness of the proposed approach, an experiment was conducted on a software engineering course at a university in Taiwan to investigate the following research questions.

(1) Do the students who learn software engineering with the flipped-classroom learning and diagnosis approach show better learning achievement than those who learn software engineering with the traditional-classroom learning approach?

(2) Do the students who learn software engineering with the flipped-classroom learning and diagnosis approach show better learning attitude than those who learn software engineering with the traditional-classroom learning approach?

(3) Do the students who learn software engineering with the flipped-classroom learning and diagnosis approach show better problem solving ability than those who learn software engineering with the traditional-classroom learning approach?

(4) What are the students' perceptions of the proposed system in terms of perceived usefulness?

3. Smart Learning Diagnosis System

To apply the flipped-classroom pedagogy to software engineering courses, this study developed a smart learning diagnosis system to assist instructors and students in conducting learning and diagnostic activities in this learning mode. Moreover, to enable the instructors and students to use various devices to operate the system, this study adopted responsive web design (RWD) technology to develop a cross-platform web application for the proposed system. The instructors and students can thus apply web browsers supported by any devices to use the system, as shown in Fig. 1.

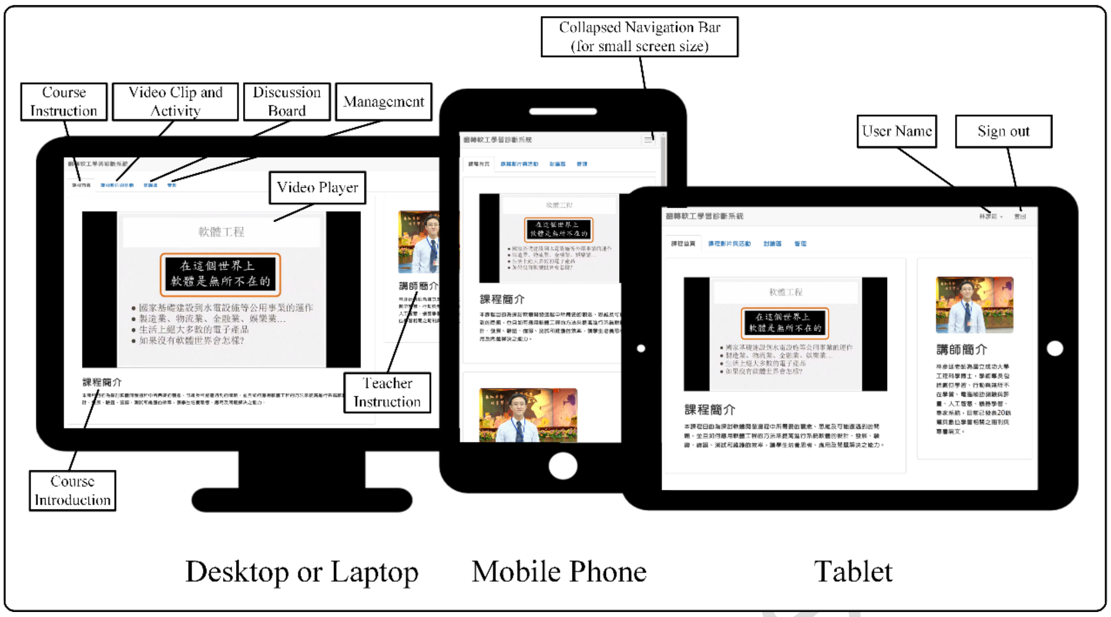

Fig. 1. Interfaces of the smart learning diagnosis system on various devices

The proposed system was implemented using PHP, and a database was built using MySQL. Fig. 2 shows the architecture of the proposed system. The system is composed of three subsystems: a self-learning system, a diagnostic system, and a management system. The proposed system can support instructors and students in the flipped classroom to conduct relevant learning activities in and out of class. With regard to the management system, instructors can use the subsystem to manage the learning activities of the software engineering course out of class. Regarding the self-learning system, students can use the subsystem to watch learning videos of the software engineering course out of class. With regard to the diagnostic system, students can use the subsystem to take diagnostic assessments to evaluate their learning status out of class.

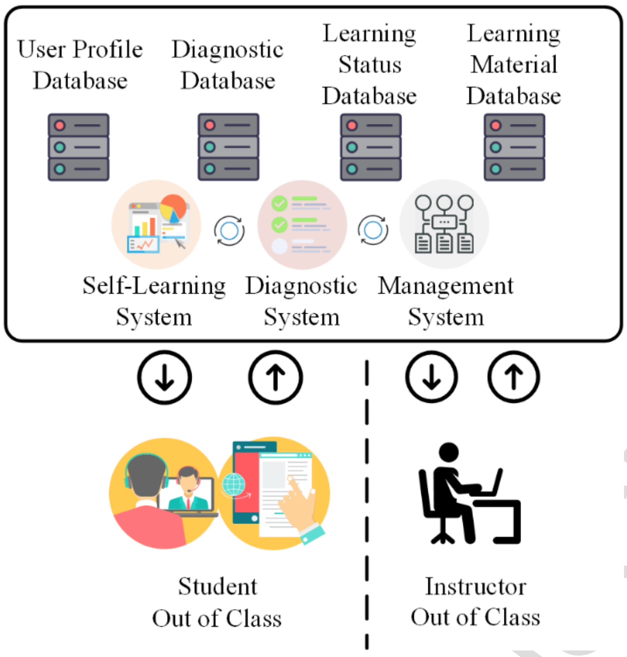

Fig. 2. The architecture of the smart learning diagnosis system

To support the flipped classroom, instructors can apply the management system to manage learning resources and diagnostic assessments for the software engineering course. As shown in Fig. 3, instructors can upload learning videos, add assessment items, and modify video descriptions through the management interfaces.

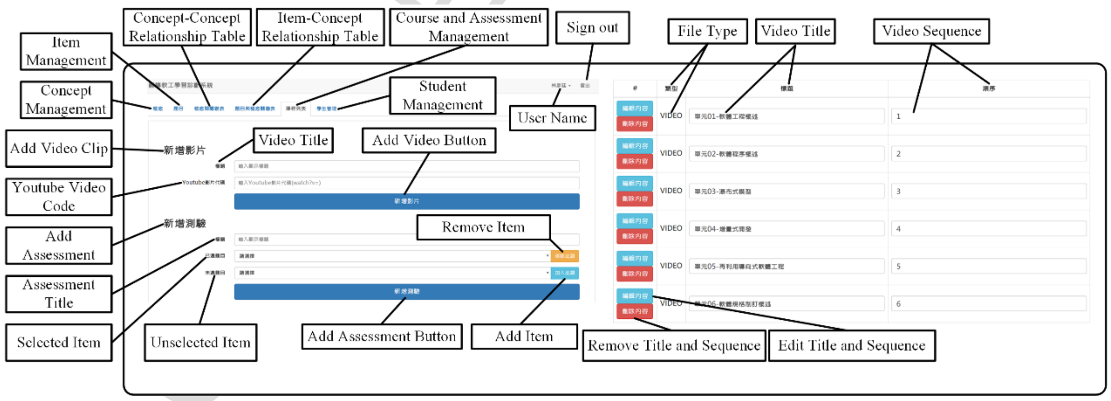

Fig. 3. Snapshots of the management interfaces

Students can use personal learning devices with the Internet to login to the system through web browsers. As shown in Fig. 4, students can view various learning resources and watch learning videos to learn the theoretical concepts of the software engineering course through the self-learning system.

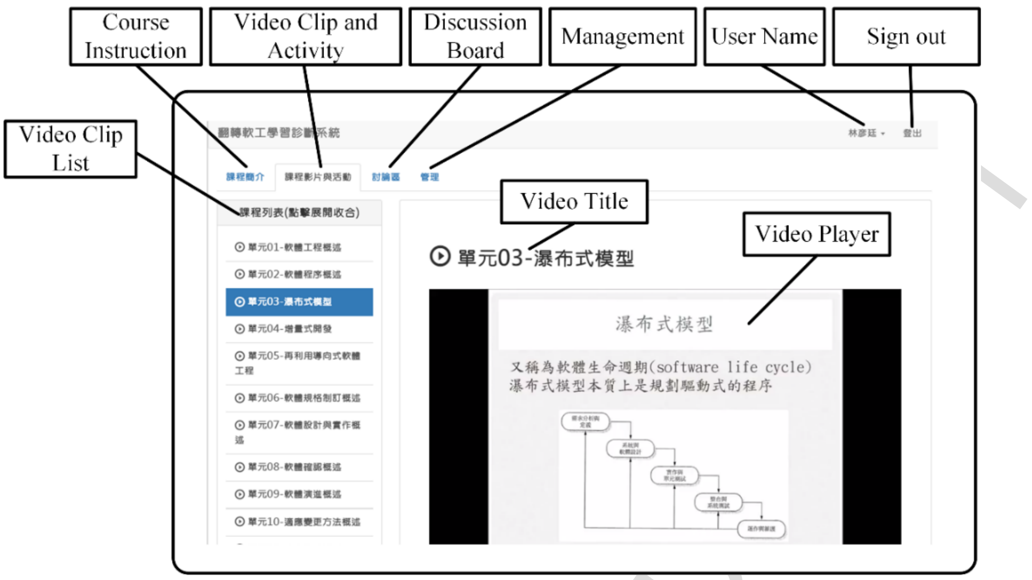

Fig. 4. Screenshot of the self-learning interface.

Furthermore, students can take diagnostic assessments to evaluate their learning status. Based on the assessment logs, the diagnostic system would be triggered to diagnose the students' learning problems and further show the diagnostic results on an individual dashboard, as shown in Fig. 5.

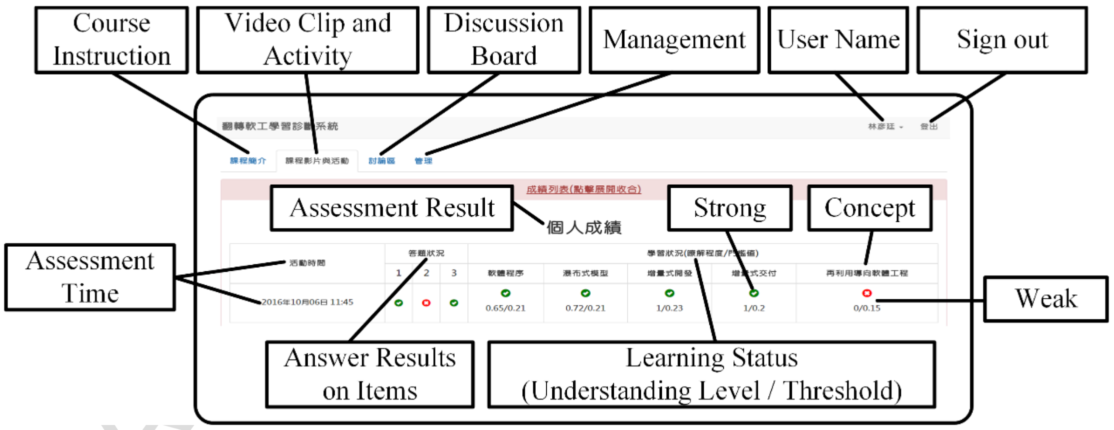

Fig. 5. Screenshot of the diagnostic result interface.

In addition, instructors can capture students' learning status and diagnostic results from the system. In this study, the diagnostic system was developed based on the testing-based approach proposed by Lin, Lin, and Huang (2011). The detailed formulation of the diagnostic system is presented in the Appendix.

As mentioned above, to store each learning resource and log, a corresponding database of the proposed system was deployed, including a learning material database, a diagnostic database, a user profile database, and a learning status database. The learning material database is a collection of software engineering video clips that includes instructional slides, annotations, and voice. The diagnostic database includes several pieces of information, such as item information, the relationship between items and concepts. The user profile database stores personal profiles that include students and instructors. The learning status database contains the learning and diagnostic status of individual students.

4. Experiment

This study aims to adopt the flipped classroom as a teaching strategy to help instructors and students to conduct a software engineering course through the use of the proposed system.

4.1. Subject

To determine whether the proposed approach truly enhances student learning performance in software engineering, a quasi-experiment was conducted on a software engineering course at a Taiwanese university. The subject of this experiment was conducted on the software development process, including requirement analysis, design, implementation, testing, and evolution. The course had a length of 10 weeks (25 hours). A total of thirty-four students and an instructor from the department of computer science asked to participate in this experiment. One group of fifteen students served as the control group. The other group of nineteen students served as the experimental group. The experimental group was supported by the flipped-classroom strategy with the proposed system to conduct the course, while the control group was supported by the traditional-classroom strategy without the proposed system. All of the students were taught by the same instructor who had taught that particular software development course for more than ten years. The students in the two groups were not aware of the intervention in this experiment.

4.2. Research instruments

To evaluate the effect of the proposed approach on student learning performance, various data sources were analyzed, including a prior knowledge test, a learning achievement test, and questionnaire results. The prior knowledge test was designed to assess the students' knowledge level with regard to software engineering before participating in the course. The learning achievement test was designed to evaluate the students' learning results after the conclusion of the course. In this study, two instructors were asked to develop the two tests together; they had taught the course more than 10 years. The two tests included 10 multiple-choice test items, and the maximum score of the tests was 100 points. Moreover, three questionnaires were adopted to measure the students' learning motivation, learning attitude, and problem solving ability. Furthermore, a system usefulness questionnaire was used to capture the perceptions of the experimental group with regard to the usefulness of the proposed system.

With regard to the investigations of the students' learning motivation and learning attitude, two questionnaires were used from the intrinsic scale of motivated strategies for learning questionnaire (MSLQ) and the learning attitude questionnaire. The learning motivation questionnaire and the learning attitude questionnaire consisted of nine items with a seven-point Likert scale and seven items with a six-point Likert scale, respectively. The two questionnaires were investigated by several studies on various courses (Hwang & Chang, 2011; Hwang, Wu, & Ke, 2011; Lin, Wen, Jou, & Wu, 2014; Pintrich & De Groot, 1990; Wei, Lin, & Lin, 2016). The learning motivation questionnaire was used to measure students' intrinsic interest in ("I think what we are learning in this class is interesting") and perceived importance of the course work ("It is important for me to learn what is being taught in this class") as well as their preferences regarding challenges and mastery of goals ("I prefer class work that is challenging so I can learn new things"). The learning attitude questionnaire was used to measure students' learning attitudes toward learning activities (e.g., "The course is valuable and worth studying" and "I would like to know more about the learning targets").

To measure the students' perceptions of problem solving ability and system usefulness, two questionnaires were used from the problem solving ability questionnaire and the perceived usefulness scale of the technology acceptance model (TAM). The questionnaire for problem solving ability included 25 items with a five-point Likert scale, and the questionnaire for the perceived usefulness of the proposed system included five items with a seven-point Likert scale (Davis, Bagozzi, & Warshaw, 1989; Lin, Lin, Huang, & Cheng, 2013; Liu, Chen, Sun, Wible, & Kuo, 2010). The problem-solving ability questionnaire was used to measure students' problem-solving attitudes toward the software development process (e.g., "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 system usefulness questionnaire was used to measure students' belief that the technology will improve their performance (e.g., "I could improve my learning performance by using this system" and "I think using this system helps me learn").

4.3. Experimental procedures

Fig. 6 shows the experimental process. Students in the experimental group and control group were asked to take four pretests before undergoing the software engineering learning activities. The first three pretests were conducted to capture the initial learning motivation, learning attitude, and problem solving ability of the two groups by using the learning motivation questionnaire, learning attitude questionnaire, and problem solving ability questionnaire. The fourth pretest was the prior knowledge test to evaluate the level of the students' background knowledge with regard to software engineering.

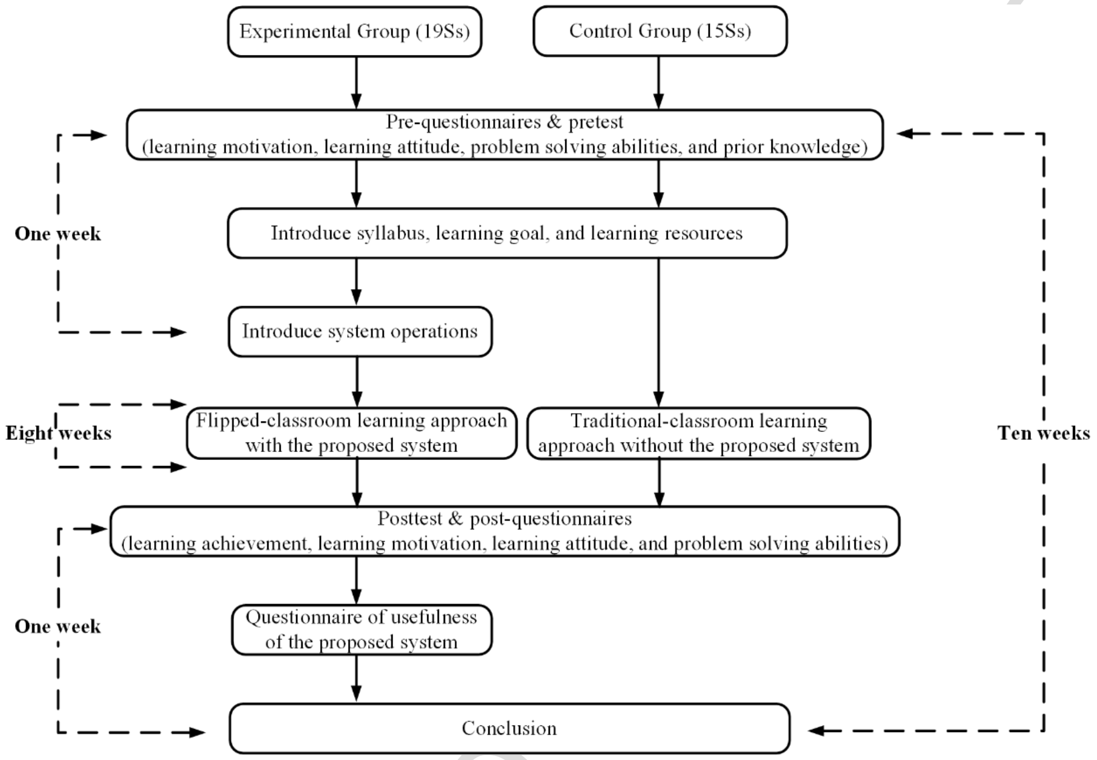
Experimental Group (19Ss)
Control Group (15Ss)
Pre-questionnaires & pretest
(learning motivation, learning attitude, problem solving abilities, and prior knowledge)
One week
Introduce syllabus, learning goal, and learning resources
Introduce system operations
Eight weeks
Flipped-classroom learning approach with the proposed system
Traditional-classroom learning approach without the proposed system
Ten weeks
Posttest & post-questionnaires
(learning achievement, learning motivation, learning attitude, and problem solving abilities)
One week
Questionnaire of usefulness of the proposed system
Conclusion

Fig. 6. The experimental process

Before engaging in the formal learning activities, the students in the control group and experimental group first received 30-min of instruction with regard to the traditional-classroom learning strategy and flipped-classroom learning strategy in the software engineering course, respectively. Furthermore, the students in the experimental group obtained an additional 20-min of instruction with regard to the operations of the proposed system.

During the course session, the instructor instructed the students in the control group in theoretical concepts of software engineering by using slides in class. Moreover, case studies, discussions, and practice exercises were also used to facilitate the students' high-order thinking by utilizing the remaining time in class. Out of class, the students in the control group were asked to complete two open-ended questions (e.g., "Please explain how to apply the agile method to speedup software development and deploy it") with regard to software engineering every week.

With regard to the experimental group, the students were asked to engage in the flipped classroom. Out of class, the students were asked to engage in self-learning to learn the theoretical concepts of software engineering by watching 61 video clips on the proposed system. Table 2 shows the information in the video clips with regard to the software development process. Moreover, the students made specific diagnosis assessments to evaluate their level of understanding through the proposed system. In class, the instructor facilitated the students' engagement in case studies, discussions, and practice activities. Furthermore, the instructor assigned two open-ended questions to the students in the experimental group every week. In addition, during the course session, the instructor applied the proposed system to diagnose the students' learning problems and further discussed the diagnostic results with the students to promote their learning.

Table 2
The information on the video clips in the software engineering course.

#	Unit	Duration	#	Unit	Duration
01	Introduction to Software Engineering	07:42	32	Structural Models Part 2	04:49
02	Introduction to Software Processes	04:59	33	Behavioral Models Part 1	04:46
03	Waterfall Model	07:24	34	Behavioral Models Part 2	05:47
04	Incremental Development	06:16	35	Introduction to Design and Implementation	02:11

05	Reuse-Oriented Software Engineering	05:32	36	Object-oriented design using the UML Part 1	02:57
06	Introduction to Software Specification	05:24	37	Object-oriented design using the UML Part 2	04:46
07	Introduction to Software Design and Implementation	03:07	38	Object-oriented design using the UML Part 3	03:05
08	Introduction to Software Validation	03:44	39	Object-oriented design using the UML Part 4	02:45
09	Introduction to Software Evolution	01:34	40	Object-oriented design using the UML Part 5	06:38
10	Introduction to Cope 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	Incremental delivery	05:03	43	Introduction to Software Testing Part 1	05:11
13	Introduction to Agile Software Development	06:19	44	Introduction to Software Testing Part 2	05:25
14	Agile Methods	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	Introduction to 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	Introduction to 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	Introduction to 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 going through all of the learning activities, all the students from the two groups received four posttests and completed the learning motivation questionnaire, learning attitude questionnaire, and problem solving ability questionnaire. The fourth posttest was the learning achievement test with regard to the software development process they learned in the course. Furthermore, the students in the experimental group were asked to complete the perceived usefulness questionnaire to survey their perceptions with regard to the usefulness of the proposed system.

5. Results

The IBM SPSS was applied to analyze the performance of the students in the experiment, including the results of the prior knowledge test, learning achievement test, learning motivation questionnaire, learning attitude questionnaire, problem solving ability questionnaire, and usefulness of the proposed system questionnaire.

5.1. Analyses of prior knowledge and learning achievement

To measure the students' prior knowledge and learning achievement, two tests were conducted before and after the software engineering learning activities.

With regard to the prior knowledge test, the mean value and standard deviation of the test scores were 48.00 and 14.73 for the control group and 36.84 and 17.33 for the experimental group. To evaluate the equivalent of the students' background knowledge with regard to software engineering before participating in the learning activities, an independent sample *t*-test was applied to analyze the prior knowledge test results between the two groups. Before the analysis, a Shapiro-Wilk test was used to examine the normality of the above data since the participating students constituted less than 50 samples in the experimental group and control group. The value of this test was 0.953 ($p > 0.05$), indicating that the sample satisfied the assumption of normality.

Furthermore, Levene's test for equality of variances was statistically insignificant ($F(1,32) = 2.02$, $p = 0.656 > 0.05$), which indicates that the group variances could be treated as equal. To further check the *t*-test result, it reveals that there were no significant differences between the experimental group and the control group ($t(1,32) = 1.988$, $p = 0.055 > 0.05$). In addition, the effect size (d) of the prior knowledge test was 0.69, representing a moderate effect size (Cohen, 1988). The result implies that the students' prior knowledge with regard to software engineering in both groups was statistically equivalent before undergoing the course.

To investigate the effectiveness of the proposed approach for improving the learning achievement of the students in the software engineering course, a one-way independent sample analysis of covariance (ANCOVA) was used to exclude the difference between the prior knowledge of the two groups. To conduct the ANCOVA, the learning achievement and prior knowledge test scores were treated as the dependent variable and covariate, respectively, and the homogeneity of the regression coefficient was tested first. The result confirmed the homogeneity of the regression coefficient ($F(1,32) = 0.204$, $p > 0.05$). Table 3 shows the ANCOVA results of the learning achievement for the two groups. The adjusted means and standard deviations were 84.22 and 3.40 for the experimental group and 72.65 and 3.85 for the control group. There was a statistically significant difference between the adjusted means ($F(1,31) = 4.779$, $p = 0.036 < 0.05$). Moreover, the learning achievement of the experimental group was significantly higher than that of the control group. In addition, the effect size (η^2) of the learning achievement test was 0.134, representing a large effect size (Cohen, 1988). The result reveals that the flipped classroom software engineering course with the proposed system benefits students more than the traditional classroom software engineering course without the system in terms of learning achievement.

Table 3
The ANCOVA results for the students' learning achievement.

Group	Number of students	Mean	S.D.	Adjusted mean	Adjusted S.D.	*F*(1,31)	*p*-value
Experimental Group	19	83.68	8.95	84.22	3.40	4.779	0.036*
Control Group	15	73.33	19.15	72.65	3.85		

Note: S.D.: Standard deviation.
*$p < 0.05$

5.2. Analyses of learning motivation and learning attitude

With regard to the analysis of the learning motivation, all the participating students were asked to complete the learning motivation questionnaire before and after the learning activities to evaluate their learning motivation. The pretest and posttest Cronbach's alpha values of the questionnaire were 0.913 and 0.909, respectively. To explore whether there were any significant differences between the means of the learning motivation of the two groups after engaging in the entire learning process, an ANCOVA was used to exclude this difference between the pretest of the learning motivation of the two groups, with the posttest and pretest scores of the learning motivation treated as the dependent variable and covariate, respectively. The homogeneity of the regression coefficient was not violated ($F(1,32) = 0.005$, $p = 0.942 > 0.05$). Table 4 shows the ANCOVA results of the learning motivation for the two groups. The adjusted means and standard deviations were 5.77 and 0.12 for the experimental group and 5.35 and 0.13 for the control group. There was a statistically significant difference between the adjusted means ($F(1,31) = 5.818$, $p = 0.022 < 0.05$). In addition, the effect size (η^2) of the posttest of the learning motivation was 0.157, representing a large effect size (Cohen, 1988). The result implies that the proposed approach significantly benefits students in terms of learning motivation.

Table 4
The ANCOVA results for the students' learning motivation.

Group	Number of students	Mean	S.D.	Adjusted mean	Adjusted S.D.	*F*(1,31)	*p*-value
Experimental Group	19	5.72	0.66	5.77	0.12	5.818	0.022*
Control Group	15	5.41	0.63	5.35	0.13		

Note: S.D.: Standard deviation.
$^{*}p < 0.05$

With regard to the analysis of the learning attitude, all the participants in the two groups were asked to complete the learning attitude questionnaire before and after taking the software engineering course. The pretest and posttest Cronbach's alpha values of the questionnaire were 0.834 and 0.852, respectively. To evaluate the equivalent of the students' learning attitudes with regard to the software engineering course before participating in the learning activities, an independent sample *t*-test was applied to analyze the pretest scores of the learning attitude between the two groups. Before the analysis, a Shapiro-Wilk test was used to examine the normality of the above data since the participating students constituted less than 50 samples in the experimental group and control group. The value of this test was 0.910 ($p = 0.073 > .05$), indicating that the sample satisfied the assumption of normality. Furthermore, a Levene's test for equality of variances was statistically insignificant ($F(1,32) = 0.287$, $p = 0.596 > 0.05$), which indicates that the group variances could be treated as equal. To further check the *t*-test result, there was no significant difference between the experimental group and the control group ($t(1,32) = -0.340$, $p = 0.736 > .05$). In addition, the effect size (d) of the pretest of the learning attitude was 0.17, representing a small effect size (Cohen, 1988). The result indicates that the two groups of students had an equivalent awareness of their learning attitudes before entering the course.

In addition, an independent sample *t*-test was performed on the rating scores to compare the posttest scores of the learning attitudes between the two groups. The values of the Shapiro-Wilk test and Levene's test were 0.916 ($p = 0.095 > .05$) and 1.960 ($p = 0.171 > 0.05$), indicating that the sample satisfied the assumption of normality and homogeneity. The analysis result reveals that there was no significant difference in the posttest scores of learning attitude between the two groups ($t(1,32) = 0.818$, $p = 0.419 > .05$). In addition, the effect size (d) of the posttest of the learning attitude was 0.27, representing a small effect size (Cohen, 1988). The result reveals that the students' learning attitudes with regard to the software engineering course in both groups was also statistically equivalent after undergoing the course.

To further investigate the students' learning attitudes, a paired sample *t*-test was used to examine the difference in the learning attitudes for the two groups before and after the learning process. Table 5 shows that there was a significant difference between the students' posttest and pretest scores for learning attitude in the experimental group ($t(1,18) = 3.899$, $p = 0.001 < 0.05$). In addition, with regard to the control group, the result shows that there was no significant difference in the students' learning attitudes before and after participating in the learning process ($t(1,14) = 1.015$, $p = 0.327 > 0.05$). Therefore, it can be seen that the flipped classroom software engineering course with the proposed system significantly benefits students in terms of learning attitude.

Table 5
The paired *t*-test results of the learning attitude for the two groups.

Group	Tests	Number of students	Mean	S.D.	*t*	*p*-value
Experimental group	Posttest	19	3.330	0.412	3.899	0.001*
	Pretest	19	3.045	0.316		
Control group	Posttest	15	3.200	0.560	1.015	0.327
	Pretest	15	3.085	0.381		

Note: S.D.: Standard deviation.
$^{*}p < 0.05$

5.3. Analyses of problem solving ability and system usefulness

With regard to the analysis of problem solving ability, all the students in the two groups were asked to complete the problem solving ability questionnaire before and after the learning activities to evaluate their problem solving ability. The pretest and posttest Cronbach's alpha values of the questionnaire were 0.769 and 0.780, respectively. To investigate the difference in the problem solving ability between the two groups after engaging in the course, an ANCOVA was used to exclude this difference between the pretest of the problem solving ability of the two groups, with the posttest and pretest scores of the problem solving ability treated as the dependent variable and covariate, respectively. The regression coefficient analysis revealed that the assumption of homogeneity was supported by the ANCOVA ($F(1,32) = 0.585$, $p > 0.05$). Table 6 shows the ANCOVA results of the problem solving ability for the two groups. The adjusted means and standard deviations were 3.732 and 0.062 for the experimental group and 3.521 and 0.072 for the control group. There was a statistically significant difference between the adjusted means ($F(1,31) = 4.855$, $p = 0.035 < 0.05$). Moreover, the problem solving ability of the experimental group was significantly higher than that of the control group. In addition, the effect size (η^2) of the posttest of learning attitudes was 0.139, representing a large effect size (Cohen, 1988). The result reveals that the flipped classroom software engineering course with the proposed system benefits students more than the traditional classroom software engineering course without the system with regard to problem solving ability.

Table 6
The ANCOVA results for the students' problem solving ability.

Group	Number of students	Mean	S.D.	Adjusted mean	Adjusted S.D.	*F*(1,31)	*p*-value
Experimental Group	19	3.730	0.315	3.732	0.062	4.855	0.035*
Control Group	15	3.522	0.170	3.521	0.072		

Note: S.D.: Standard deviation.
$^{*}p < 0.05$

To evaluate the perceptions of the experimental group with regard to the usefulness of the proposed system, the perceived usefulness questionnaire was used and revised based on the TAM. The Cronbach's alpha value of the questionnaire was 0.908, and the results of the investigation are shown in Table 7. The investigation results show that 98.9% of the students perceived the usefulness of the proposed system.

Table 7
The results of the students' perceptions of using the proposed system in the experimental group.

#	Question	EU (%)	QU (%)	SU (%)	Neither (%)	SL (%)	QL (%)	EL (%)	Mean
1	Using the proposed system in the flipped classroom software engineering course would enable me to learn and diagnose relevant knowledge more effectively	0	0	0	0	10.5	52.6	36.8	6.26
2	Using the proposed system would improve	0	0	0	0	10.5	57.9	31.6	6.21

	my learning performance in the flipped classroom software engineering course								
3	Using the proposed system in the flipped classroom software engineering course would increase my learning comprehension productivity	0	0	0	0	15.8	57.9	26.3	6.10
4	Using the proposed system would make it easier to learn software engineering in the flipped classroom	0	0	0	5.3	10.5	36.9	47.4	6.26
5	I would find the proposed system useful in the flipped classroom software engineering course	0	0	0	0	10.5	57.9	31.6	6.21

In addition, a multivariate analysis of covariance (MANCOVA) was used to have a statistical control of the pre-existing difference in this study. The pretest scores of learning achievement, learning motivation, learning attitude, and problem solving ability were set as covariates to analyze the difference between the pretest and posttest scores for all of the dependent variables. As shown in Table 8, the results indicate that the posttest scores for learning achievement, learning motivation, learning attitude, and problem solving ability differed significantly between the two groups (*Wilks' Λ* = 0.597, $F = 4.056$, $p = 0.012$). The result reveals that the flipped classroom software engineering course with the proposed system benefits students more than the traditional classroom software engineering course without the system in terms of learning achievement, learning motivation, learning attitude, and problem solving ability.

Table 8
The MANCOVA results for learning achievement, learning motivation, learning attitude, and problem solving ability of the control group and experimental group.

Effect	*Wilk's Λ*	*F*	Hypothesis *df*	Error *df*	η^2	Observed Power[a]	*p*-value
Group	0.597	4.056	4	25	0.403	0.847	0.012*

[a] Computed using alpha = 0.05; $^*p < 0.05$

6. Discussion and conclusions

This study proposed a flipped classroom with a smart learning diagnosis system to support a software engineering course. Moreover, an experiment was conducted to evaluate the effectiveness of the proposed approach. The experimental results showed that, in comparison with the traditional-classroom learning approach, the proposed approach significantly improved the students' learning achievement, learning motivation, and learning attitude. Furthermore, the students who learned with the proposed approach had stronger problem solving abilities than those who learned with the traditional-classroom learning approach. In addition, most students in the experimental group agreed on the usefulness of the proposed system in the flipped classroom software engineering course.

These findings provide evidence that the proposed approach can benefit students in terms of software engineering learning. From the aspect of learning achievement, the proposed system applied in the proposed approach provides a strong learning and diagnosis tool for instructors and students since appropriate learning and assessment activities have a significant effect on learning achievement in a flipped classroom (Wang, 2017). From the perspective of learning motivation, this study applied the RWD technique to develop a cross-platform application to facilitate content delivery in the flipped classroom. This design was consistent with past research findings, which noted that the effective application of technology in a flipped classroom is an important indicator of students' learning motivation (Bergmann & Sams, 2012; Davies, Dean, & Ball, 2013). Moreover, the literature indicates that sufficient prior knowledge can also enhance students' learning motivation and achievement (Lai & Hwang, 2016; Lin, Lin, & Huang, 2011). In addition, with regard to learning attitude, past investigations indicated that the development of intelligent techniques in an online learning platform for a flipped classroom would positively affect students' learning attitude (Zhai, Gu, Liu, Liang, & Tsai, 2017). This point was also consistent with the diagnostic tool of the proposed system in this study. In terms of learning activity design, several studies have noted the specific advantages that could be achieved by conducting appropriate diagnostic activities during the learning process (Huang, Huang, & Wu, 2014; Hwang, Panjaburee, Triampo, & Shih, 2013; Panjaburees, Triampo, Hwang, Chuedoung, & Triampo, 2013). Furthermore, the literature indicates that it is preferable to implement a flipped-classroom approach in a small class (< 20 students) since it is possible to involve all the students in class activities at one time (Galway, Corbett, Takaro, Tairyan, & Frank, 2014; Kerr, 2015).

Overall, the major contribution of this study is to propose a flipped classroom with a smart learning diagnosis system to support software engineering education. Based on the proposed approach, this study has some limitations and opportunities for future work. In the present study, the experimental results can only indicate that the flipped classroom with the proposed system can benefit student learning performance in a software engineering course. The result does not fully reflect the impact of the proposed system on student learning performance. Therefore, to address this issue, a further experiment should be conducted to investigate the student learning performance of a software engineering course between a traditional classroom without the proposed system, a flipped classroom without the proposed system, and a flipped classroom with the proposed system and further investigate the effect of the system on student learning performance in the flipped classroom. Moreover, as the sample size of the experiment was not large, this study needs to continuously conduct more software engineering courses to cover various samples and provide additional evidence. To facilitate human-computer interactions in the flipped classroom, a modern technique, called chatbot, should be integrated with the proposed system to provide intelligent learning services for instructors and students.

Appendix

The diagnostic system was developed to evaluate students' theoretical concepts of software engineering while learning with the proposed system. It assumes that a student takes a diagnostic assessment and the assessment involves several items, and the student has to apply corresponding concepts to answer the items in the assessment. According to the context, three data relationships from the assessment were defined: the relationship between the items and the concepts, the relationship among the concepts, and the relationship between the student's answers and the items, as shown in Tables A.1, A.2, and A.3. Moreover, three variables were used to represent the three relationships, *X*, *Z*, and *R*. The values of *X* and *Z* ranged from 0 to 1, indicating the relevant degree from weak to strong. In addition, the value of *R* is set to 0 or 1, which indicates the correctness of the student's answer on each item.

Table A.1
The relationship between the items and the concepts.

Item	Concept						
	C_1	C_2	C_3	…	C_i	…	C_n
I_1	X_{11}	X_{21}	X_{31}	…	X_{i1}	…	X_{n1}
I_2	X_{12}	X_{22}	X_{32}	…	X_{i2}	…	X_{n2}
I_3	X_{13}	X_{23}	X_{33}	…	X_{i3}	…	X_{n3}
…	…	…	…	…	…	…	…

I_j	X_{1j}	X_{2j}	X_{3j}	…	X_{ij}	…	X_{nj}
…	…	…	…	…	…	…	…
I_k	X_{1k}	X_{2k}	X_{3k}	…	X_{ik}	…	X_{nk}

Table A.2
The relationship among concepts.

Concept	Concept						
	C_1	C_2	C_3	…	C_i	…	C_n
C_1	Z_{11}	Z_{21}	Z_{31}	…	Z_{i1}	…	Z_{n1}
C_2	Z_{12}	Z_{22}	Z_{32}	…	Z_{i2}	…	Z_{n2}
C_3	Z_{13}	Z_{23}	Z_{33}	…	Z_{i3}	…	Z_{n3}
…	…	…	…	…	…	…	…
C_m	Z_{1m}	Z_{2m}	Z_{3m}	…	Z_{im}	…	Z_{nm}
…	…	…	…	…	…	…	…
C_n	Z_{1n}	Z_{2n}	Z_{3n}	…	Z_{in}	…	Z_{nn}

Table A.3
The relationship between students' answers and the items.

Item	Student						
	S_1	S_2	S_3	…	S_l	…	S_r
I_1	R_{11}	R_{21}	R_{31}	…	R_{l1}	…	R_{r1}
I_2	R_{12}	R_{22}	R_{32}	…	R_{l2}	…	R_{r2}
I_3	R_{13}	R_{23}	R_{33}	…	R_{l3}	…	R_{r3}
…	…	…	…	…	…	…	…
I_j	R_{1j}	R_{2j}	R_{3j}	…	R_{lj}	…	R_{rj}
…	…	…	…	…	…	…	…
I_k	R_{1k}	R_{2k}	R_{3k}	…	R_{lk}	…	R_{rk}

Based on the definitions, the importance of each concept in a diagnostic assessment is measured as Eq. (A.1).

$$\mathrm{CI}(C_i) = \frac{\sum_{m=1}^{n}\sum_{j=1}^{k} Z_{im}X_{mj}}{\sum_{i=1}^{n}\sum_{m=1}^{n}\sum_{j=1}^{k} Z_{im}X_{mj}} \tag{A.1}$$

where CI(C_i) represents the importance of the ith concept in the diagnostic assessment, $0 \leqq \mathrm{CI}(C_i) \leqq 1$; Z_{im} indicates the relevant degree between ith and mth concepts, $0 \leqq Z_{im} \leqq 1$; and X_{mj} represents the relationship between the mth concept and jth item, $0 \leqq X_{mj} \leqq 1$.

Moreover, based on the relationship between the student's answers and the items, the understanding level of the student with regard to the concepts in the diagnostic assessment can be inferred as Eq. (A.2)

$$\mathrm{UL}(S_l, C_i) = \frac{\sum_{m=1}^{n}\sum_{j=1}^{k} R_{lj}Z_{im}X_{mj}}{\sum_{i=1}^{n}\sum_{m=1}^{n}\sum_{j=1}^{k} Z_{im}X_{mj}} \tag{A.2}$$

where UL(S_l, C_i) represents the understanding level of the lth student on the ith concept, $0 \leqq \mathrm{UL}(S_l, C_i) \leqq 1$; R_{lj} represents the correctness of answer of the lth student on the jth item; Z_{im} indicates the relevant degree between ith and mth concepts; and X_{mj} represents the relationship between the mth concept and jth item, $0 \leqq X_{mj} \leqq 1$.

In addition, to further evaluate whether the student has a sufficient level of understanding of the concepts, Eq. (A.3) was formulated based on the investigation in Khumalo (2006).

$$\mathrm{T}(C_i) = m \times CI(C_i) + b \tag{A.3}$$

where T(C_i) represents the threshold value of the ith concept, $0 \leqq \mathrm{T}(C_i) \leqq 1$; m indicates the gradient of the function, m=1; CI(C_i) represents the importance of the ith concept in the diagnostic assessment, $0 \leqq \mathrm{CI}(C_i) \leqq 1$; and b is the point at which the line crosses the y-axis, b=0.

Therefore, the diagnostic system can infer the concepts with regard to which the student is weak through the above functions.

References

Andreessen, M. (2011). Why software is eating the world. *The Wall Street Journal*.

http://online.wsj.com/news/articles/SB10001424053111903480904576512250915629460. Accessed July 17, 2017.

Baeten, M., Kyndt, E., Struyven, K., & Dochy, F. (2010). Using student-centred learning environments to stimulate deep approaches to learning: Factors encouraging or discouraging their effectiveness. *Educational Research Review*, 5(3), 243-260.

Baker, A., Navarro, E. O., & van der Hoek, A. (2005). An experimental card game for teaching software engineering processes. *Journal of Systems and Software*, 75(1-2), 3-16.

Bergmann, J., & Sams, A. (2012). *Flip your classroom: Reach every student in every class every day*. Eugene, OR: International Society for Technology in Education.

Bergmann, J., & Sams, A. (2014). *Flipped learning: Gateway to student engagement*. Washington, DC: International Society for Technology in Education.

Bransford, J., Sherwood, R., Vye, N., & Rieser, J. (1986). Teaching thinking and problem solving: Research foundations. *American psychologist*, 41(10), 1078.

Chamillard, A. T., & Braun, K. A. (2002). The software engineering capstone: structure and tradeoff. *ACM SIGCSE*, 34(1), 227–231

Chen, C. Y., & Teng, K. C. (2011). The design and development of a computerized tool support for conducting senior projects in software engineering education. *Computers & Education*, 56(3), 802-817.

Chen, Y. L., Wang, Y. P., Kinshuk, & Chen, N. S. (2014). Is FLIP enough? Or should we use the FLIPPED model instead?. *Computers & Education*, 79, 16–27.

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences (2nd ed.)*. Hillsdale, NJ: Erlbaum.

Davis, F., Bagozzi, R., & Warshaw, R. (1989). User acceptance of computer technology: A comparison of two theoretical models. *Management Science*, 35(8), 982–1003.

Davies, R. S., Dean, D. L., & Ball, N. (2013). Flipping the classroom and instructional technology integration in a college-level information systems spreadsheet course.

Educational Technology Research and Development, 61(4), 563-580.

Galway, L. P., Corbett, K. K., Takaro, T. K., Tairyan, K., & Frank, E. (2014). A novel integration of online and flipped classroom instructional models in public health higher education. B*MC Medical Education*, 14, 181.

Hadjerrouit, S. (2005). Learner-centered web-based instruction in software engineering. *IEEE Transactions on Education*, 48(1), 99-104.

Hainey, T., Connolly, T. M., Stansfield, M., & Boyle, E. A. (2011). Evaluation of a game to teach requirements collection and analysis in software engineering at tertiary education. *Computers & Education*, 56(1), 21-35.

Heppner, P. P., & Petersen, C. H. (1982). The development and implications of a personal problem-solving inventory. *Journal of Counseling Psychology*, 29(1), 66.

Huang, Y. M., Huang, S. H., & Wu, T. T. (2014). Embedding diagnostic mechanisms in a digital game for learning mathematics. *Educational Technology Research and Development*, 62(2), 187-207.

Hwang, G. J., & Chang, H. F. (2011). A formative assessment-based mobile learning approach to improving the learning attitudes and achievements of students. *Computers & Education*, 56(4), 1023-1031.

Hwang, G. J., Panjaburee, P., Triampo, W., & Shih, B. Y. (2013). A group decision approach to developing concept–effect models for diagnosing student learning problems in mathematics. *British Journal of Educational Technology*, 44(3), 453-468.

Hwang, G. J., Wu, P. H., & Ke, H. R. (2011). An interactive concept map approach to supporting mobile learning activities for natural science courses. *Computers & Education*, 57(4), 2272-2280.

Jeannette M. W. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.

Jonassen, D. H. (2000). Toward a design theory of problem solving. *Educational Technology Research and Development*, 48(4), 63-85.

Jou, M, Lin, Y. T., & Tsai, H. C. (2016). Mobile APP for motivation to learning: an engineering case. *Interactive Learning Environments*, 24(8), 2048-2057.

Kerr, B. (2015). The flipped classroom in engineering education: A survey of the research. In *Paper presented at the 2015 International Conference on Interactive Collaborative Learning Proceedings, Florence, Italy*.

Khumalo, B. (2006). The fundamental theory of knowledge. *MPRA Paper*, 3733.

Kim, M. K., Kim, S. M., Khera, O., & Getman, J. (2014). The Experience of three flipped classrooms in an urban university: An Exploration of design principles. *The Internet and Higher Education*, 22, 37-50.

Kong, S. C. (2015). An Experience of a three-year study on the development of critical thinking skills in flipped secondary classrooms with pedagogical and technological support. *Computers & Education*, 89, 16-31.

Lai, C. L., & Hwang, G. J. (2016). A self-regulated flipped classroom approach to improving students' learning performance in a mathematics course. *Computers & Education*, 100, 126-140.

Lin, Y. T. (2016). *When Mobile Technology Meets Traditional Classroom Learning Environment: How Does it Improve Students' Learning Performances?*, In K. Wallace (Eds.), Learning Environments: Emerging Theories, Applications and Future Directions (Chapter 8). Nova Science Publishers, Inc.

Lin, Y. C., & Huang, Y. M. (2013). A Fuzzy-based Prior Knowledge Diagnostic Model with Multiple Attribute Evaluation. *Educational Technology & Society*, 16 (2), 119–136.

Lin, H. C., & Hwang, G. J. (2018a). Research trends of flipped classroom studies for medical courses: a review of journal publications from 2008 to 2017 based on the technology-enhanced learning model. *Interactive Learning Environments*, doi.org/10.1080/10494820.2018.1467462.

Lin, C. J., & Hwang, G. J. (2018b). A Learning Analytics Approach to Investigating

Factors Affecting EFL Students' Oral Performance in a Flipped Classroom. *Educational Technology & Society*, 21 (2), 205–219.

Lin, Y. T., Lin, Y. C. (2017). *Applying Mobile Technology to the Support Learning and Diagnosis Approach in a Flipped Classroom*, In D. René and C. Aubin (Eds.), Mobile Learning: Students' Perspectives, Applications and Challenges (Chapter 5). Nova Science Publishers, Inc.

Lin, Y. C., Lin, Y. T., & Huang, Y. M. (2011). Development of a diagnostic system using a testing-based approach for strengthening student prior knowledge. *Computers & Education*, 57(2), 1557-1570.

Lin, Y. T., Lin, Y. C., Huang, Y. M., & Cheng, S. C. (2013). A wiki-based teaching material development environment with enhanced particle swarm optimization. *Educational Technology & Society*, 16(2), 103–118.

Lin, Y. T., Wen, M. L., Jou, M., & Wu, D. W. (2014). A cloud-based learning environment for developing student reflection abilities. *Computers in Human Behavior*, 32, 244-252.

Liu, I. F., Chen, M. C., Sun, Y. S., Wible, D., & Kuo, C. H. (2010). Extending the TAM model to explore the factors that affect intention to use an online learning community. *Computers & Education*, 54(2), 600–610.

Maratou, V., Chatzidaki, E., & Xenos, M. (2016). Enhance learning on software project management through a role-play game in a virtual world. *Interactive Learning Environments*, 24(4), 897-915.

Meyer, A. N., Fritz, T., Murphy, G. C., & Zimmermann, T. (2014). Software developers' perceptions of productivity. In *Paper presented at the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering Proceedings, Hong Kong, China*.

Moreno, A., Sanchez-Segura, M., Medina-Dominguez, F., & Carvajal, L. (2012). Balancing software engineering education and industrial needs. *The Journal of Systems* *and Software*, 85(7), 1607-1620.

Panjaburees, P., Triampo, W., Hwang, G. J., Chuedoung, M., & Triampo, D. (2013). Development of a diagnostic and remedial learning system based on an enhanced concept effect model. *Innovations in Education and Teaching International*, 50(1), 72-84.

Pintrich, P. R., & De Groot, E. V. (1990). Motivational and self-regulated learning components of classroom academic performance. *Journal of Educational Psychology*, 82(1), 33–40.

Saiedian, H. (2002). Bridging Academic Software Engineering Education and Industrial Needs. *Computer Science Education*, 12(1-2), 5-9.

Schultz, D., Duffield, S., Rasmussen, S. C., & Wageman, J. (2014). Effects of the flipped classroom model on student performance for advanced placement high school chemistry students. *Journal of Chemical Education*, 91(9), 1334-1339.

Slomanson, W. R. (2014). Blended learning: A Flipped classroom experiment. *Journal of Legal Education*, 64(1), 93-102.

Sommerville, I. (2010). *Software Engineering (9th Edition)*. Boston, Addison-Wesley.

Stross, R. (2012). *Computer Science for the Rest of Us*. The New York Times, BU5.

Teo, T. W., Tan, K. C. D., Yan, Y. K., Teo, Y. C., & Yeo, L. W. (2014). How flip teaching supports undergraduate chemistry laboratory learning. *Chemistry Education Research and Practice*, 15(4), 550-567.

Voogt, J., & Roblin, N. P. (2012). A comparative analysis of international frameworks for 21st century competences: Implications for national curriculum policies. *Journal of Curriculum Studies*, 44(3), 299-321.

Wang, F. H. (2017). An exploration of online behaviour engagement and achievement in flipped classroom supported by learning management system. *Computers & Education*, 114, 79-91.

Wei, C. W., Lin, Y. C., & Lin, Y. T. (2016). An interactive diagnosis approach for supporting clinical nursing courses. *Interactive Learning Environments*, 24(8), 1795-1811.

Zhai, X., Gu, J., Liu, H., Liang, J. C., & Tsai, C. C. (2017). An Experiential Learning Perspective on Students' Satisfaction Model in a Flipped Classroom Context. *Educational Technology & Society*, 20 (1), 198–210.

This study is supported by the Ministry of Science and Technology, Taiwan, R.O.C. under grants MOST 104-2511-S-153-002-MY2 and MOST 106-2511-S-153-003-MY2.

- This study proposed a flipped classroom with a smart learning diagnosis system.
- An experiment was conducted in a software engineering course.
- The proposed approach is helpful to students in improving learning performance.
- Most students showed positive perceptions toward the usage of the proposed system.