International Journal of Adult Education and Technology
Volume 14 • Issue 1

Designing an Effective Teaching Learning Environment for an Undergraduate Software Engineering Course

A. M. Abirami, Thiagarajar College of Engineering, India*
S. Pudumalar, Thiagarajar College of Engineering, India
S. Thiruchadai Pandeeswari, Thiagarajar College of Engineering, India

ABSTRACT

Software engineering is a core theory course offered in undergraduate engineering programmes which deals with various systematic approaches, methods, and tools that can be employed for designing, developing, testing, and maintaining quality software applications. It is one of the challenging courses for the teaching faculty. After graduation, the students are expected to practice software engineering principles and practices in their prospective projects. In order to satisfy the industrial requirement, and to create industry ready software professionals, software engineering course is offered as theory cum practical course. This article focuses on setting up an effective learning environment for learning software engineering courses. The course is blended with traditional classroom teaching methodology along with a set of pedagogical practices, active learning strategies, ICT tools for content delivery and assessment. This enhanced approach in teaching learning methodology improves student learning outcomes, which in turn helps them to adopt corporate practices more easily.

KEYWORDS

Active Learning Strategies, Assessment, Course Outcomes, ICT tools, Learning Environment, Pedagogical Practices, SDLC Practices, Software Engineering

INTRODUCTION

By definition, Software Engineering refers to the process of designing, developing and testing applications/software by collecting and analyzing user requirements by following well-defined principles, standards and guidelines. It is customary that Software Engineering, as a course that deals with the scientific procedures, principles and standards which are followed and incorporated to develop quality software is imparted to undergraduate Engineering students of Computer Science and Information Technology domains. The Software Engineering course normally aims to present various standard methods, tools and procedures available for quality software development. The course

DOI: 10.4018/IJAET.321655 *Corresponding Author emphasizes systematic and quantifiable approaches available for the development and maintenance of software. It helps the students to identify a real time problem, analyze and understand the requirements, formulate functional and non-functional requirements, choose a suitable process model for the development of software and incorporate the milestones prescribed in the process model appropriately. Further students get to learn the umbrella activities of project development such as project scheduling, staffing, various standard project and product metrics. Covering a rich set of contents, this course is paramount for Undergraduate Engineering Students to prepare them for their future prospects as a Software Engineering Professional. This paper presents various technology enhanced pedagogical practices adopted in the content delivery and assessment of Software Engineering Course offered to the students of Information Technology Engineering programme. Adopting the suggested practices proved to create an effective learning environment for students as the majority of students were able to achieve all of the course outcomes.

BACKGROUND

The Software Engineering course is offered at the third semester along with Data structures and Object Oriented Programming courses. At this point in time of their Undergraduate degree course, students begin learning the basics of computer systems and leverage them to solve problems. Learners are expected to learn the various aspects of software development and develop applications that suit real needs. Hence it is thought to be ideal to offer the Software Engineering course at this juncture, so that students could develop and test their projects incorporating the standard procedures, guidelines, tools, methods, coding standards, project management aspects and ethical practices imparted as part of the course.

The course was initially offered based on the traditional pedagogical strategy of following the lectures with problem solving activities in and out of the classroom. However, the following shortcomings were found in the traditional content delivery method:

- Limited in-class and in-lab activities restricted the scope of experimentation by the learners, as there were no well-defined guidelines to carry out off-the-class activities such as requirements gathering and feasibility analysis
- Lack of adaptation of technology assisted methods to monitor the progress made in off-the-class activities
- Lack of rubrics to assess how well the learners carried out the offline activities such as client meetings, requirements gatherings, feasibility analysis for developing the application, excluding code development
- There was very less scope for improving and assessing interpersonal skills of learners while they work in teams offline
- Lack of Rubrics to assess the managerial and interpersonal skills of learners

The above-mentioned factors led to poor attainment of course outcomes. Hence suitable improvements in the pedagogical strategies were considered for adoption.

At the first step, the course was re-designed as a Theory cum practical course with 24 hours of lecture and 24 hours of hands-on sessions in the lab. Theory-cum-practical course design facilitated a systematic set-up for allowing students to do hands-on practice. It also helped the course handler to monitor the students progress in real-time and facilitated both providing and receiving necessary feedback for improvement. Course plan for the content delivery of the course was laid out in such a way that the 24 lecture hours would be used for delivering software engineering concepts that are aligned with suitable active learning methods. Students spend the remaining 24 hours in the lab where they practice incorporating the software engineering concepts in the projects they develop.

Further, to develop an effective teaching learning environment a broad study on the recommended methods and practices proposed in existing literature was made. The observations and inference made through the study are presented below.

Chen et. al. (2009) emphasized that to prepare students for industrial-scale software development with traditional teaching methodology would not work well. Only hands-on experience could provide better understanding of the methods and concepts of software engineering. The paper explored the usage of four approaches such as group work, experiment, case study and cooperative education in software engineering education and showed how these approaches upgraded their teaching learning process. To make industry ready students, the most important generic attributes that the students should develop during their studies are teamwork, presentation and communication skills, learning by experimentation and case study that can be inculcated using these types of approaches.

Venson et. al. (2016) presented a framework of academy-industry collaboration that would facilitate students to undergo practical and academic activities in a real world scenario. The research was based on elements outside the classroom that could prepare the students for professional practices. An IDEAL instructional model *(Initiating, Diagnosing, Establishing, Acting, and Learning)* was used for developing the curriculum of Software Engineering. Under this study, students were asked to solve the real time problems given by the Ministry of Telecommunications (MC), Brazil. Students practiced various agile process models to solve the stakeholder's problems. The impact was students were able to make real time products and publish their work in conferences.

Alabbadi et. al. (2016) proposed novel methods by combining cooperative learning and mastery learning strategies to teach software engineering using social media tools. The paper highlighted the challenges that the students face due to lack of appropriate skills when beginning their career. The author emphasized that usage of modern ICT tools and teaching strategies in classrooms would support students to actively learn Software Engineering than traditional chalk and talk approach. Office Software Engineering Services (OSES) was proposed to facilitate the faculty and students with industry trends. The various social media tools suggested for teaching and learning software engineering are Wikis, Blogs, Micro blogs, Tagging, Podcast.

Huan (2012) explored practice-based teaching methods for Software Engineering course by analyzing the shortcomings of traditional methods of teaching. The author claimed that Software Engineering course was artificially divided as Software Engineering system and the Software Engineering practice. The author suggested that case-based learning and creating a space for students to discuss and practice software methodology concepts in the classroom would really help the students to understand what Software Engineering is all about.

Ramos et. al. (2018) suggested that to fulfil the software industry requirements, professionals with skills for collaborative teamwork, data analysis and problem solving is possible by adopting active learning methodologies in classrooms. The author experimented and presented the impact of using Team based learning for Software Engineering course. Team Based Learning (TBL) along with IDEAL approach showed increase in attainment of course objectives. However, lack of tool support for some specific features of the TBL methodology, led to the specification of a customized tool for the context and teaching environment has been encountered.

Ramachandran (2017) emphasized primary course requirements like clear learning objectives, learning outcome, proposed assessment, and clear marking criteria in order to have an efficient teaching and learning model. The author proposed Active Learning & Teaching model based on specific learning outcomes *(ALT Model)* by enforcing three types of learning practices such as Divergent Thinking, Differential Assessment and Collaborative Learning. The approach was implemented with second year and final year students where the final year students guided the second year students and collaborated with them. The ALT model was also evaluated by students and found to be 100% more effective in their learning.

Alramouni et. al. (2018) conducted a study from academic year 2013 to 2015*(2 terms in each year)*for Software Engineering course by investigating various strategies for identifying at-risk students

(students those who have a higher probability of failing academically or rare class participation), engaging and making positive attention of those students through interesting class activities, appropriate assessment methods and assignments to make connection between their classroom learning and real-world problems. Team projects on real world problems showed improvement in their learning and achievements.

Fonseca et. al. (2017) explored Problem-based Learning (PBL) and agile software engineering practices in order to achieve learning outcomes and to improve students' participation during classes. The main objectives of their experiment was to bring students closer to their professional development, exposing students to a real project from the outset until achieving a software application by applying software development methodologies. SCRUM methodology was used to guide the software construction. Each team performed an oral presentation of their project. The learning achievements of two academic year students were compared in terms of their pass rates and student's participation in activities.

Effective learning environment and blended mode for teaching learning process suits for all types of courses. Hubscher et. al (2003) used new pedagogical approaches like constructive and collaborative activities and visual demonstration for learning algorithms which improved self learning among the students. Liu et. al. (2013) adopted blended approach to teach Data Structures and Algorithms course. The faculty used this model for teaching methods and experiments through visual demonstrations, project based teaching, eLearning, and so on. Maguire et. al. (2014) adopted pair programming model to teach Computer Programming, Data Structures, and so on for the students who had not studied computer science courses in their school days. Continuous assessment was carried out in different aspects like interdependence and co-ordination between the pair, and individual contribution to problem solving which resulted in drastic reduction of overall failure rate. Yang et. al. (2016) adopted ARCS motivation model for teaching Data Structures course. The assessment was carried out using Pair-programming model, and each student was asked to comment on the other's program implementation of a Data Structure concept. This paper proposed yet another collaboration technique to improve students' performance in the course Data Structures involving various teaching styles through ALS techniques for the content delivery.

Eison (2010) studied the importance of ALS inside the classroom and to create learning environment to have enhanced learning. Malmi et. al (2008) proposed active learning strategies and examination methods for learning Data Structures and Algorithms using internet, small-scale demonstrations and simulations. Students are asked to design solutions for practical applications. Cummings et. al. (2017) proposed different blended mode strategies for teaching which would advance skill sets of students through learner-centred instructions along with tools and technologies. Pandey (2012) proposed a collaborative and interactive learning platform for Uttarakhand schools (India) for learning mathematics (geometry) and science using open source tools. Nail et. al. (2017) discussed the advantages of mobile technology and gadgets inside the classroom for the improvement of learning. Bano et. al. (2018) had done a detailed study on the use of mobile apps for teaching mathematics and science and categorized them into different domain, type and context of use. Yessenova et. al. (2020) analyzed the support of e-Learning practices for the distance mode of education in Kazakhstan and provided recommendations for its further development.

From the inferences made out of the literature survey, it was decided to ensure the following to ensure an effective teaching learning environment.

- The course must have well defined learning outcomes and effective assessment methods
- The course should aim to develop generic attributes of learners like team working skills and communication skills to shape them as better software professionals
- Students must be involved in real time problem solving as part of this course
- Use of technology must be incorporated into the content delivery and assessment. Social media tools like wikis, blogs, podcasts must be integrated into content delivery to ensure pervasive learning
- Students must be given suitable platforms for communicate and collaborate among themselves
- The collaborations may be suitably monitored and mended, if needed.

MOTIVATION

There are a number of works in literature that evaluated the effectiveness of the following components in the instructional design.

- employment of modern ICT tools and Active learning strategies
- usage of group work, experimentation, case studies and cooperative education
- collaborative learning through discussion and practice both in and out of classroom
- use of Wikis, Blogs, Micro blogs, Tagging, Podcast etc.,
- engaging and making positive attention to the students who are slow on the uptake through interesting class activities, appropriate assessment methods and assignments

However, an instructional design that encompasses all the aspects addressed by the above components has not been evolved specifically for the Software Engineering course. Hence an instructional design for software Engineering course was tailored encompassing all these components and systematic assessment techniques. The evolved design includes a blended content delivery plan that consists of an array of activities to be performed in-class, in-lab and offline as a team. The activities were all monitored and assessed using well-rounded Rubrics as and when needed. It became essential to evaluate the design to ensure its effectiveness. Criteria for the evaluation and their significance needed to be fixed.

Hence RQ1 presented below is considered as the primary motivation of this study.

RQ1: Does a blended learning approach improve the learning outcomes of students? What is the impact of implementing blended learning practices?

Course outcomes based on Bloom's taxonomy are defined and are taken as high-level reference for measuring the effectiveness of the instructional design evolved. The effectiveness of the design is measured through level of attainment of course outcomes.

While the RQ1 is concerned with the cognitive domain of learning, the second important motivation of this work is to measure the response of the students for the blended content delivery plan in the affective domain. The satisfaction index is measured through surveys and formative feedback. The RQ2 presents the secondary motivation of this study.

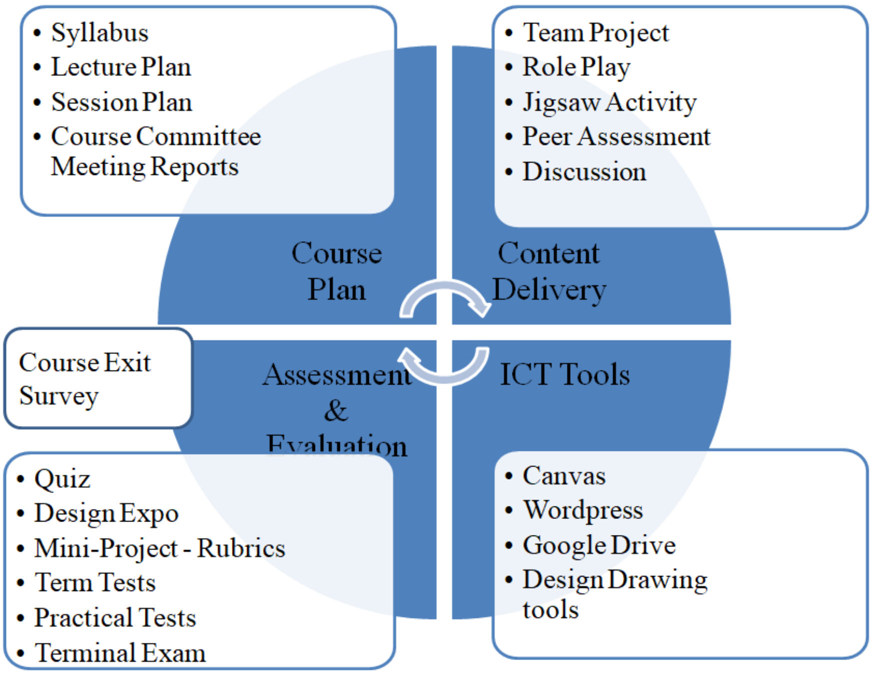
RQ2: Does a blended learning approach promote the satisfaction index and performance of students?

This research is an experimental study on implementation of blended learning approach along with active learning strategies for the course Software Engineering.

METHODOLOGY

Over the years, teaching Software Engineering with traditional methodology paved a gap to industrial need, hence blended learning has been adopted to teach Software Engineering course. Blended learning is an approach in the education domain that combines online educational materials and opportunities for interaction with traditional place-based classroom methods. Students are taught software engineering principles and are asked to practice them through their combined mini-project implementation using the agile process model. Students are instructed to apply software engineering concepts, principles and methods for application development. The framework of blended learning practices for teaching Software Engineering course is shown in Figure 1.

Figure 1.
The framework of blended learning practices



Course Plan

The Software Engineering course has the following set of Course Outcomes (COs) with Bloom's Taxonomy level at 'Apply':

CO1 - Interpret functional and non-functional requirements for a given problem
CO2 - Prepare design documents for the given requirements
CO3 - Write test cases using appropriate testing techniques for an application
CO4 - Develop application with documentation using Software Engineering processes

Course Plan was designed in such a way that students would be engaged in different types of activities pertaining to the course outcomes and for the entire duration of the course. The course plan was laid out in such a way that Students would be engaged in a variety of activities related to requirements engineering tasks such as elicitation, client meetings, field survey, and feasibility analysis as offline activities. Blended learning facilitates the course handler to share guidelines and templates required to carry out offline activities effectively. These activities are efficiently monitored and assessed using online quizzes, discussion forums and periodic reviews. The course plan is prepared to facilitate the blended learning mode with the use of ICT tools and leveraged in Content Delivery and Assessment.

Content Delivery as Team Activities

The following are the different set of activities planned for the course Software Engineering:

i. Team Formation
ii. Problem Identification

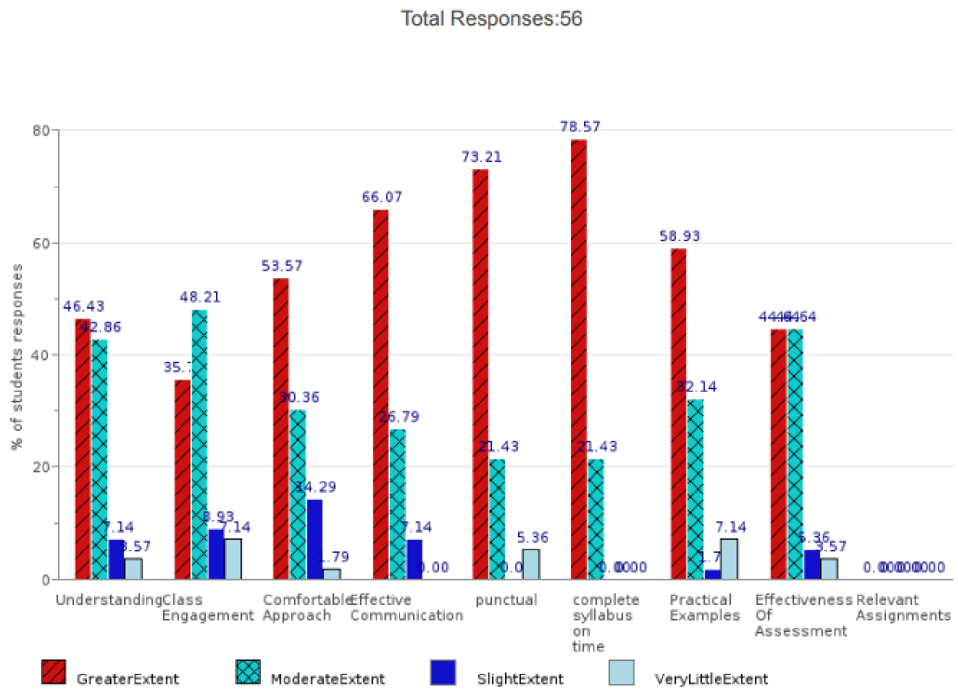
iii. Requirements Collection
iv. Design Documentation
v. Development and Testing
vi. Report and Documentation

Team formation is done with the mix of fast and slow learners. Each team comprises 3-4 members. Each student takes up roles as per agile model like *Scrum Master, Product owner, Developer, and Tester*. Students are asked to ***identify problems*** from sectors like education, health, transport, and agriculture in order to make them work on real world social or economic improvement problems. Each team has to identify their ***Client team*** among the project teams in the class. The Instructor lists out all the problem domains. Project teams having similar problem domains or the most relevant domains are assigned as Client teams for the other.

The seven tasks of Requirements Engineering like Inception, Elicitation, Elaboration, Negotiation, Specification, Validation, and Management are taught to Students with live examples and applications. The understanding of the Client's requirements collection process is done by ***Role Play.*** The team members take different roles and are enacted to describe the scenario of requirements collection from the client team *(*https://www.youtube.com/watch?v=t-4NnAhtL-0*)*.

The project teams then initiate Requirements gathering activity which includes field study, meeting client team and other stakeholders. The user requirements are collected, and discussed with client teams to get their sign off. Their journey of requirement collection process is prepared as chart presentations/videos and shared with other teams, as shown in Figure 2. Every week stand-up meetings are conducted and documented in the form of Course Journals. Project teams used powtoon and youtube for video preparations and uploads (https://www.powtoon.com/c/dAh3qMvekQd/1/m*)*.

Figure 2.
Sample chart presentation for requirements gathering



Identifying functional requirements and non-functional requirements is the challenging task for every project team. ***Software Design expo*** is conducted to overcome this issue. In this demonstration, the project teams explained their project features using their design diagrams like use case, activity, class, sequence diagrams and data flow diagrams. The faculty mentors, client team and peer learners visit each stall and validate the requirements and design documents. Project teams used different online free drawing tools for their design documentation. This type of ***Peer Assessment*** helps all the learners to have better understanding in preparing the Software Requirement Specification (SRS) Document. Sample ***Checklist*** for Peer Assessment is given in Table 1.

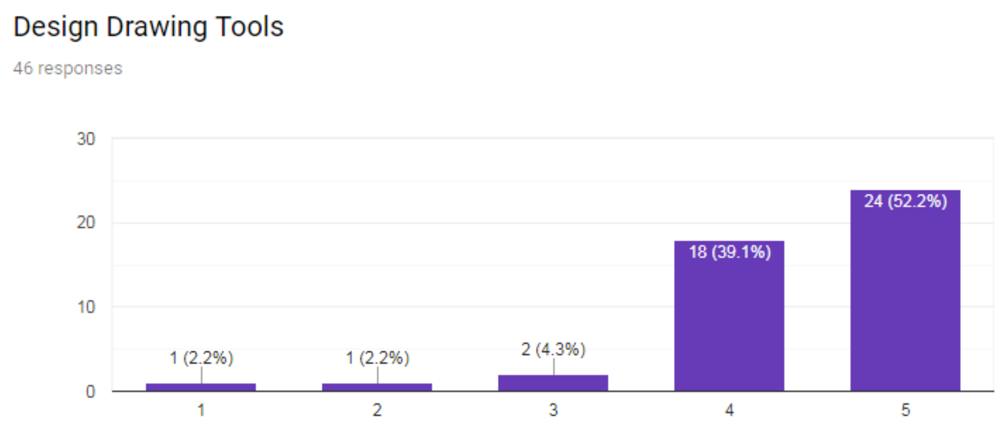
User Interface (UI) design varies from application to application like desktop, web or mobile. ***Jigsaw activity*** is done to learn UI concepts and principles behind different types of applications, as shown in Figure 3. It helps all the students to have parallel learning and improve self learning. Learners used different types of free UI tools like pencil, Wix, and so on for their UI design presentation.

Project implementation is done as part of their laboratory courses or beyond class hours. Project teams are asked to ***implement*** 2-3 requirements using any programming languages *(usually C++ or Java). Unit testing* is done by the developers. The Instructor asks the Client team to perform ***peer***

Table 1.
Sample checklist for peer assessment

Description	Yes/No	If no, write Comments
UML Diagrams		
Are all 5 UML diagrams drawn? (Usecase, Class, Activity, Sequence, StateChart)		
Has Class diagram shown all required relationships (dependency, generalization, association with multiplicity)?		
Does Classes used in Sequence diagram matched with classes used in Class diagram?		
If generalization relationship shown, is there “is-a-kind of” relationship?		
Data Flow Diagrams		
Are all 3 level DFDs drawn? (Contextual, Level 1 and Level 2)		
Is Level 0 and Level 1 or Level 1 and Level 2 same?		
Does the design document have atleast two Level 2 diagrams?		
User Interface Design		
Does the UI Design exhibit user friendliness?		
General Design Requirements		
Are software design principles/concepts followed in the design document?		
Are all UML/DFD notations, naming conventions followed in design diagrams?		
Is Drawing tool used for drawing design diagrams?		
Does this design capture all features/FRs/NFRs?		
Does this design have any ambiguities or inconsistency?		
Behavioural Skills		
Do you find any copyright violations, misuse of software, etc in the design documents?		
Has the team submitted the document for review on time?		
Are all your team members involved in this review process?		

Figure 3.
Jigsaw activity for UI design



assessment for validating the user requirements by executing the application as *User Acceptance Testing*.

Assessment and Evaluation

The assessment of Higher Order Thinking Skills of Students is assessed by project reviews, continuous assessment tests, practical test and end semester examinations (terminal examinations). Intermediate reviews are done to check the progress of project teams. Final review is supported by other Course Faculty also who are part of these mini-projects. ***Rubrics*** are used for assessing the team project which includes separate criteria for communication, as shown in Table 2 and individual Practical tests are conducted at the end of the course. Students are asked to identify requirements, draw design diagrams and write test cases for the given scenario. Student feedback is obtained using a survey tool *(in-house developed)* in two different phases: during the mid of the semester and at the end of the course. Further improvements are planned based on the analysis of student feedback reports.

Table 2.
Rubrics for mini-project implementation

Review 1 – Requirements Collection	**Review 2 – Design Documents**	**Review 3 – Verification and Validation**	**Final Review – SW Engg, OOP, DS**
Problem Selection (2)	Understanding level of Data Flow Diagrams (3)	Review on design documents (3)	Problem understanding (2)
Problem Understanding (2)	Understanding level of UML diagrams (3)	Review on Code (3)	Application of OOP concept to the problem(5)
Understanding The Functional Requirements (3)	User friendliness in User Interface Design (3)	Documentation (2)	Data Structure design for the problem (5)
Understanding the Non-Functional Requirements (3)	Communication (2)	Timeliness (1)	Documentation (2)
Communication (2)	Involvement in Team (1)		Presentation (2)
Involvement In Team (1)	Timeliness (1)		Involvement in the team (2)
Timeliness (1)			Timeliness (1)

ICT tools usage

The Instructor used ***Wordpress*** and ***Canvas tool*** as Learning Management System for sharing eResources, project documentation submissions, and ***discussions,*** as shown in Figures 4 and 5.

Instructors use the SpeedGrader feature of Canvas tool to evaluate all submissions. Course Faculty use the Quiz module of Canvas to check the understanding level of students and for sharing keys of tests. All project teams submit their presentations in Google Drive for the purpose of sharing among project teams using the document template.

RESULTS AND DISCUSSIONS

Usually, the end semester examinations measure only the knowledge level of students. The course faculty has the challenges in improving the skills and attitude levels of students. The blended learning practices along with Active Learning Strategies enable the students to acquire Higher Order Thinking Skills (HOTS).

Figure 4.
ICT Tools – Canvas

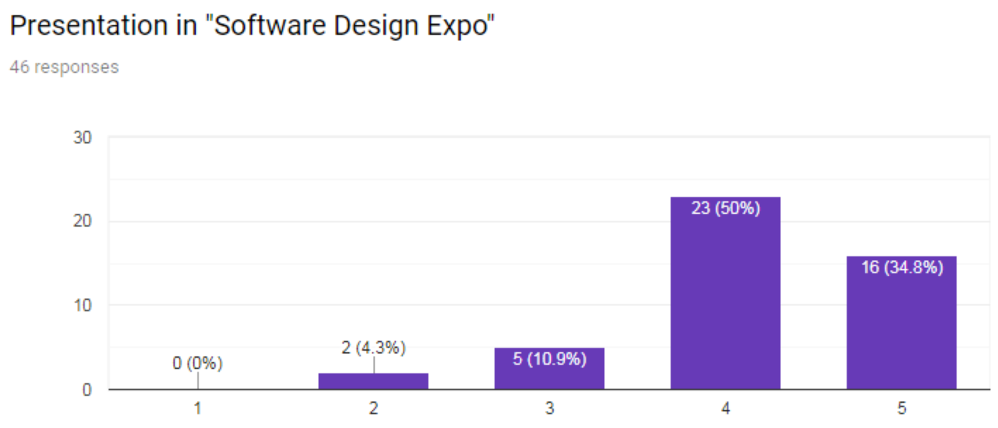
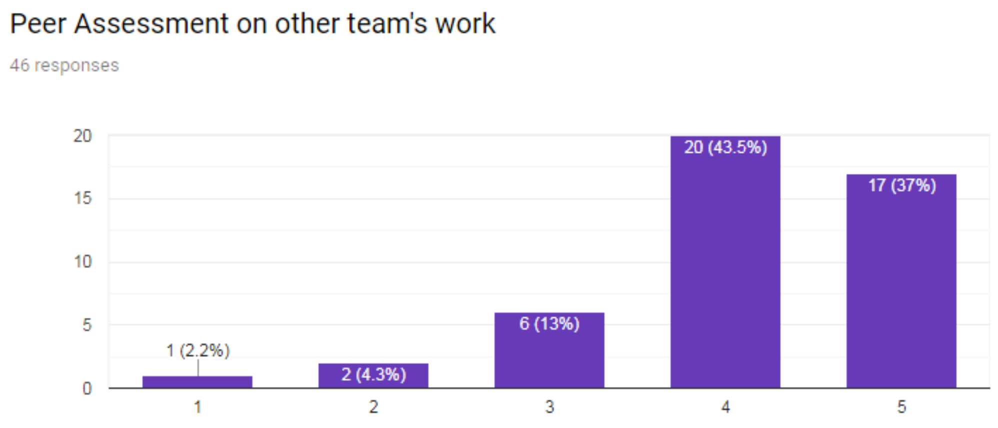


Figure 5.
ICT Tools – Wordpress



Experimentation Details

Eight different batches of students were considered for this study. Each batch consisted of 130 students on an average, for eight different academic years starting from 2011-12 to 2018-19. The student batches Batch1, Batch2, Batch3 and Batch4 were part of the academic years 2015-16, 2016-17, 2017-18 and 2018-19 respectively. The course Software Engineering was offered as the Theory cum Practical course and the instructors adopted blended learning approach. The student batches Batch1-M1, Batch1-M2, Batch1-M3 and Batch1-M4 were part of the academic years 2014-15, 2013-14, 2012-13 and 2011-12 respectively. Here, M1 says Minus-1 from the year 2015-16 where the course was offered as a theory course and the instructors followed traditional approach of teaching. These eight batches were further categorized into a control group and experimental group, based on the type of Software Engineering course delivered to them. Four batches had taken this course as a theoretical course (control group) whereas other four batches had taken this course as Theory cum Practical course (experimental group).

Attainment of Learning Outcomes

The attainment of course outcomes *(or learning outcomes)* was assessed in different modes like implementation of mini-projects, tests and end semester examinations. Data was collected from the continuous assessment tests *(3 theory tests and 1 practical test)* and the performance of students of four different batches *(from 2015-16 to 2018-19, the experimental group)* was discussed in this paper.

The students were asked to do combined mini-project implementation for three courses like Software Engineering, Object Oriented Programming and Data Structures. This ***integrated learning*** helped the students to perform better in their terminal examinations *(written mode)* and able to have a holistic understanding of the course learnt.

Table 3 discusses the student performance in Test 1 and Test 2 for the students of Batch4. Test 1 had questions from requirements collection process (CO1) and drawing UML diagrams for any given scenario (CO2); Test 2 had questions related to drawing Data Flow Diagrams (DFDs) or User Interface Design (CO2). Similarly Table 4 shows the performance of students in practical tests for CO1 and CO2.

From Tables 3 and 4, it is understood that the design thinking ability *(knowledge)* and the use of drawing tools *(skills)* by students greatly improved for Batch4. The practice of blended learning approach in the course helped in the improvement of student learning outcomes *(RQ1)*. It is evident from Figure 6 that 82% of students performed better in practical tests for drawing design diagrams, when compared with theory tests which are only 72%.

The Figures 7 and 8 show CO1 and CO2 attainment of all four batches of students. The performance of Batch3 was poor in both written and practical test, as shown in Figure 6.

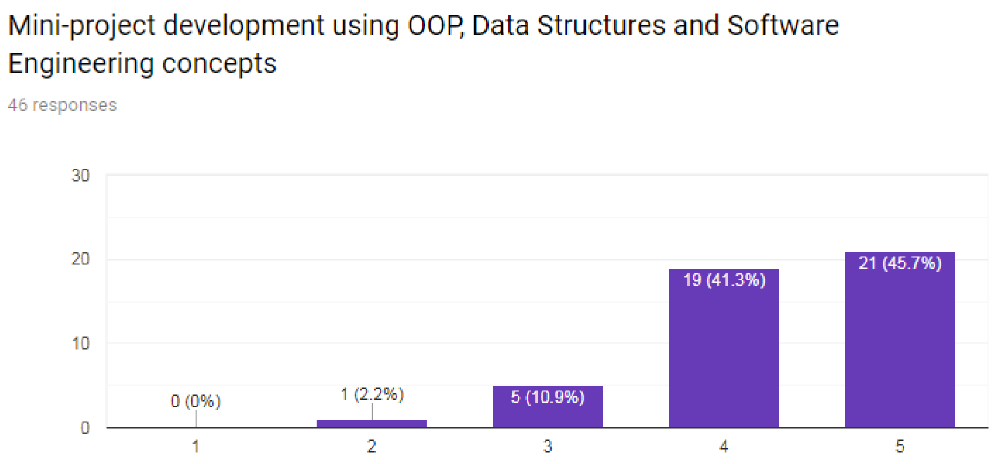
Table 3.
Student performance in written tests

Batch4 Marks Range	Number of Students		
	Test 1 (CO1 – Req)	Test 1 (CO2 - UML)	Test 2 (CO2 – DFD / UID)
>=80%	60	61	36
50-79%	39	42	53
20-49%	32	24	33
< 20%	4	8	13
Total	135	135	135

Table 4.
Student performance in practical tests

Batch4 Marks Range	Number of Students		
	CO1 – Req	CO2 – UML/DFD	CO2 –UID
>=80%	79	78	65
50-79%	49	40	37
20-49%	5	13	12
< 20%	2	4	21
Total	135	135	135

Figure 6.
Performance of students – batch 4



The Figure 8 shows that the performance of Batch1 students was better compared to Batches 2 and 3. The various factors like quality of incoming students, variation in assessment, and so on might be the reasons for deviations. The instructors took necessary measures by having more activities for the Batch4. The students of Batch4 participated in all the activities for requirements collection hence they were able to perform well in written and practical test.

Promotion of Overall Performance of Students

The adoption of blended learning approach promotes student performance *(RQ1)*. Figure 9 shows the overall performance of students of 4 different Batches for the course Software Engineering. It includes both continuous assessment and end semester examinations marks. The performance of Batch4 students improved in all aspects when compared with Batch2 and Batch3.

From Figure 9, it is evident that 75% of students scored greater than 60%; there is a gradual decrease in the percent of failures (less than 50%). The overall performance of eight batches in their end semester examinations for the course Software Engineering is shown in Table 5.

Table 5 describes the percentage of students for different ranges of scores. For example, 28% of Batch4 students scored 60-70% marks in their end semester examinations. Generally, the fast learners *(those who score 80% and above)* perform well irrespective of teaching methodology practiced. The

Figure 7.
Comparison of performance of students – CO1

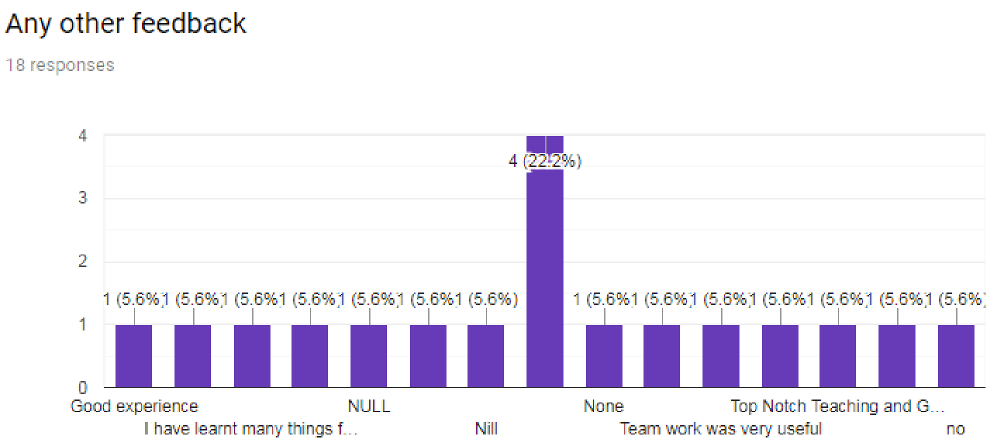
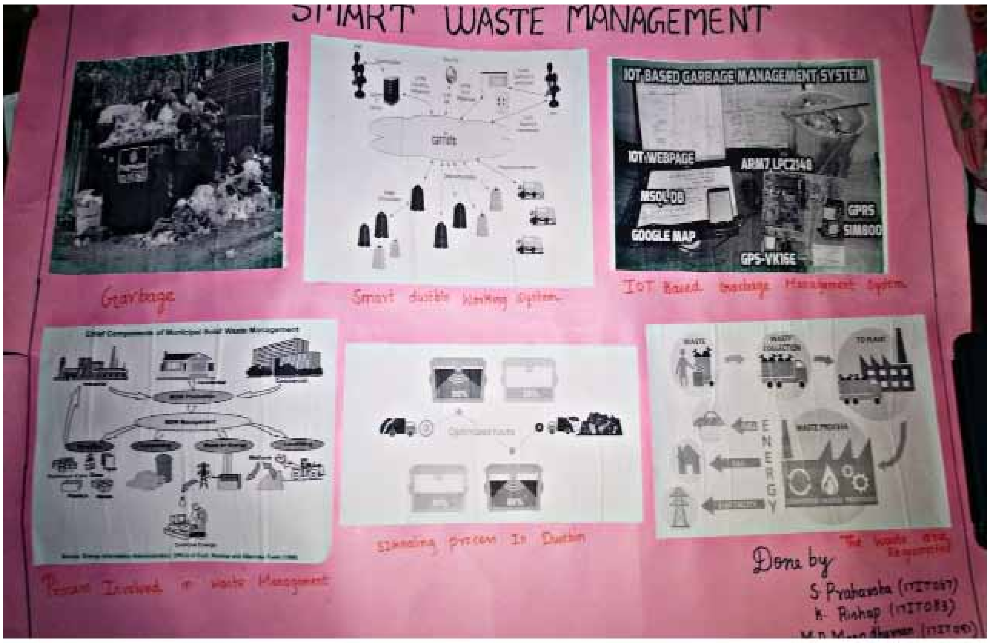


Figure 8.
Comparison of performance of students – CO2


SMART WASTE MANAGEMENT
IOT BASED GARBAGE MANAGEMENT SYSTEM
IOT WEBPAGE
ARM7 LPC2148
MSQL DB
GOOGLE MAP
GPRS SIM800
GPS-VK16E
Garbage
Smart dustbin Working System
IOT Based Garbage Management System
Persons Involved in waste Management
Sensing system In Dustbin
ENERGY
Done by
S Prahasha
K Rishap

blended approach is the most required teaching learning practice for the other type of learners *(those who score <80%)*.

The experimental group refers to the batches of students where this course was offered as the Theory cum Practical course; on the other hand, the control group refers to the batches of students where this course was offered as the Theory course. The average of students of these two different groups in different marks ranges is shown in Table 6. For example, the average of the experimental group is 17 for the set of students who scored marks in the range 50-60%.

Figure 9.
Overall performance of students

Table 5.
Overall performance of students

Score Batch	Percentage of Students			
	70-80%	60 -70%	50-60%	< 50%
Batch1- M4	24	25	10	6
Batch1- M3	22	15	7	5
Batch1- M2	20	18	8	8
Batch1- M1	28	14	10	9
Batch1	24	16	10	10
Batch2	31	24	22	8
Batch3	29	25	20	12
Batch4	32	28	16	9

Table 6.
Comparison of performance of students

Type of Students Group	Average of Batches of Students			Average
	Score 70 – 80%	Score 60 – 70%	Score 50 – 60%	
Control Group (Theory Course)	23.5	18	8.75	16.75
Experimental Group (Theory cum Practical Course)	29	23.25	17	23.08

It is evident from Table 6 that the class average has increased to 5%, 7% and 10% for the student groups of 70-80%, 60-70% and 50-60% scorers respectively, wherein these students had undergone blended mode of learning *(RQ1)*. Thus, the blended mode of teaching practices helps the students to get scores more than the required pass percentage value *(≥50%),* which in turn reduces the number of failures.

Hypothesis Testing

The effectiveness of the use of blended mode of teaching Software Engineering course is measured with the support of students' performance in this course across various batches. Hypothesis testing is done using t-test at the significance level 0.05. Null hypothesis is rejected if p-value is less than 0.05.

Null Hypothesis: The design of Software Engineering course as Theory cum Practical course and with the blended mode of teaching learning practices would improve the average of students scoring marks in the range 50-80%, $\mu>20$. Here the pass percentage in the course is considered as 50% for each student.

Alternate Hypothesis: The design of Software Engineering course as Theory cum Practical course and with the blended mode of teaching learning practices would improve the average of students scoring marks in the range 50-80%, $\mu\leq20$. Here the pass percentage in the course is considered as 50% for each student.

The t-test has been conducted for the average scores of two groups *(control and experimental)*, for the values shown in Table 6. The p-value is 0.022; the null hypothesis can be rejected and the alternate hypothesis can be accepted, as the p-value is less than 0.05.

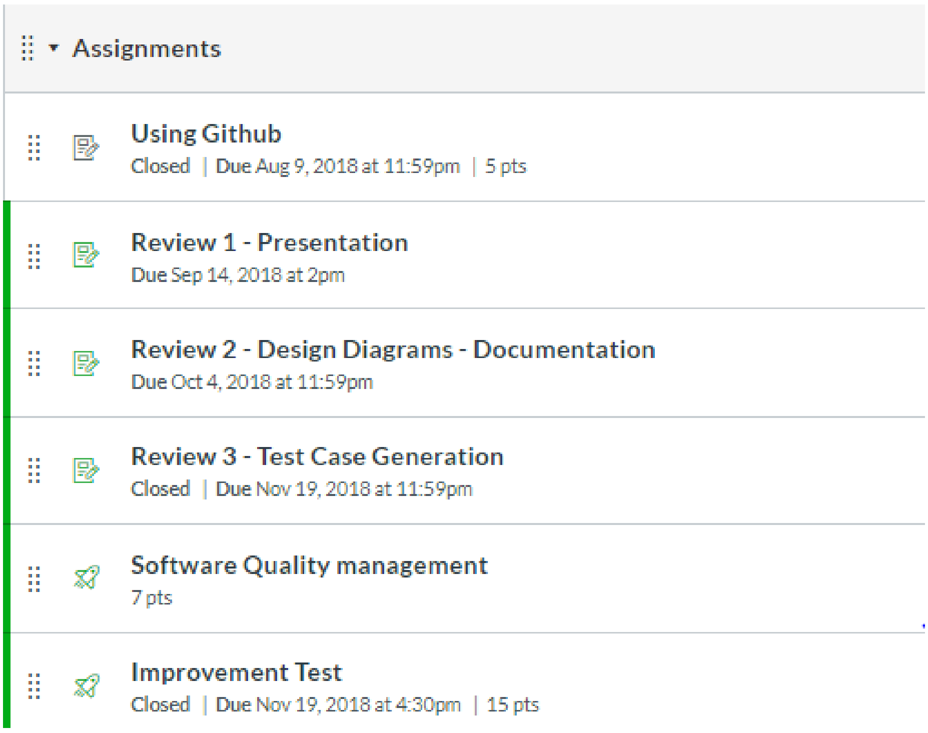
Feedback Analysis and Continuous Improvement

Figure 10 shows the mid-semester feedback of students during the course period. Mostly the content delivery methods followed by Course Faculty and the understanding level of students are assessed here.

At the end of course, students were asked to record their feedback on how much the activities like team presentation, team work, tools usage, documentation, peer team review help in their individual performance. ***Google form*** had been used to collect the feedback. Figures 11, 12, 13, 14 and 15 show the responses from 46 students, '1' being the most unlikely and '5' being the most likely.

The analysis on the textual feedback says that students overall enjoyed the course, learnt new tools and earned good experience while working in the project team. Student's attitude towards learning

Figure 10.
Mid-semester feedback analysis report



Assignments

Using Github
Closed | Due Aug 9, 2018 at 11:59pm | 5 pts

Review 1 - Presentation
Due Sep 14, 2018 at 2pm

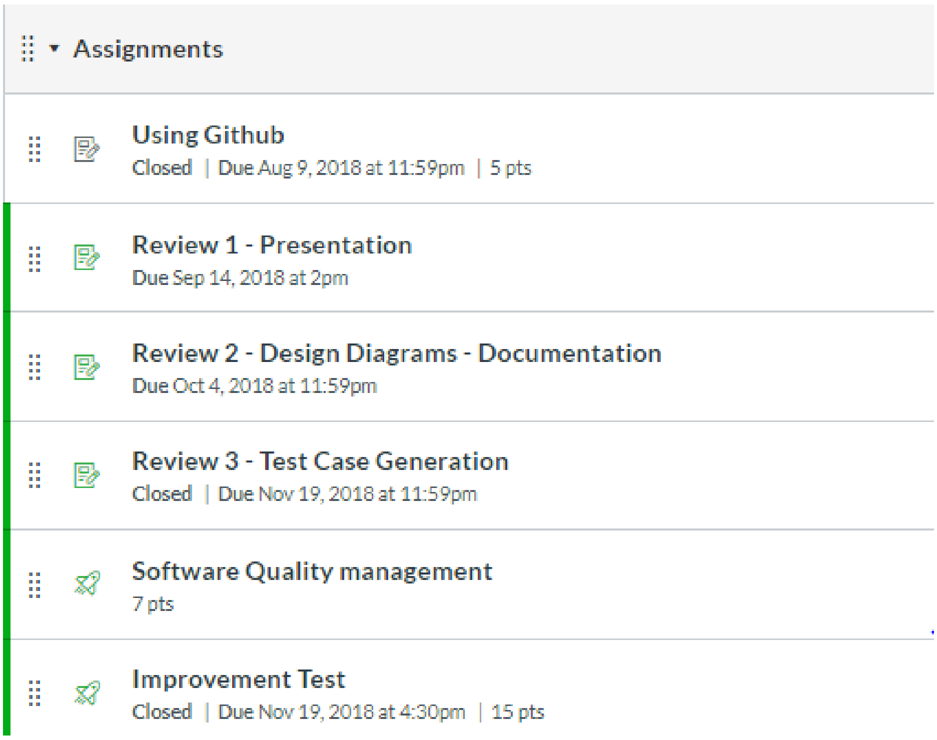
Review 2 - Design Diagrams - Documentation
Due Oct 4, 2018 at 11:59pm

Review 3 - Test Case Generation
Closed | Due Nov 19, 2018 at 11:59pm

Software Quality management
7 pts

Improvement Test
Closed | Due Nov 19, 2018 at 4:30pm | 15 pts

Figure 10.
Mid-semester feedback analysis report



Assignments

Using Github
Closed | Due Aug 9, 2018 at 11:59pm | 5 pts

Review 1 - Presentation
Due Sep 14, 2018 at 2pm

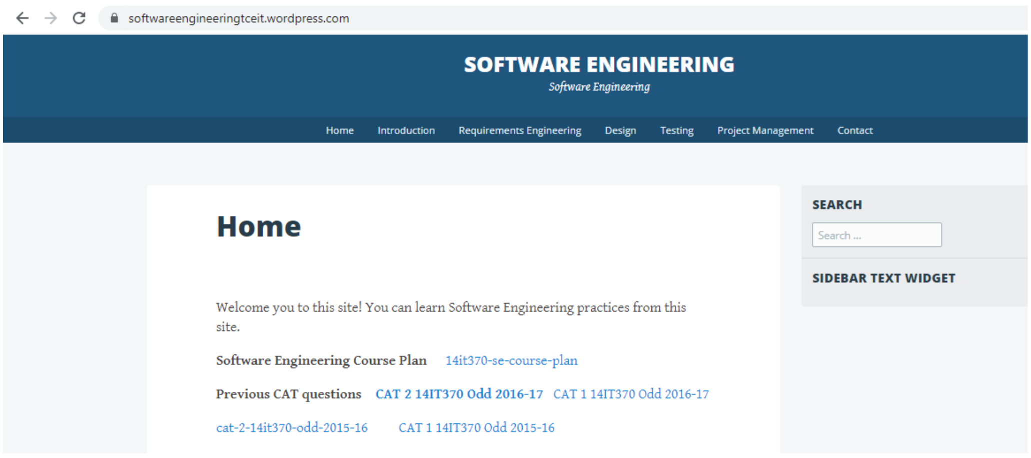
Review 2 - Design Diagrams - Documentation
Due Oct 4, 2018 at 11:59pm

Review 3 - Test Case Generation
Closed | Due Nov 19, 2018 at 11:59pm

Software Quality management
7 pts

Improvement Test
Closed | Due Nov 19, 2018 at 4:30pm | 15 pts

Figure 11.
Course end survey – sample question Q1



softwareengineeringtceit.wordpress.com

SOFTWARE ENGINEERING
Software Engineering

Home Introduction Requirements Engineering Design Testing Project Management Contact

Home

Welcome you to this site! You can learn Software Engineering practices from this site.

Software Engineering Course Plan 14it370-se-course-plan

Previous CAT questions CAT 2 14IT370 Odd 2016-17 CAT 1 14IT370 Odd 2016-17

cat-2-14it370-odd-2015-16 CAT 1 14IT370 Odd 2015-16

SEARCH
Search ...

SIDEBAR TEXT WIDGET

beyond the curriculum and lifelong learning was also developed. Hence a blended learning approach paved the way to improve KSA *(Knowledge, Skill, and Attitude)* eventually *(RQ2)*.

Students were also asked to record their feedback on the learning outcomes in the likert scale, 1 being the lowest rating and 4 being the highest rating. The satisfaction index for the course outcomes

Figure 12. Course end survey – sample question Q2

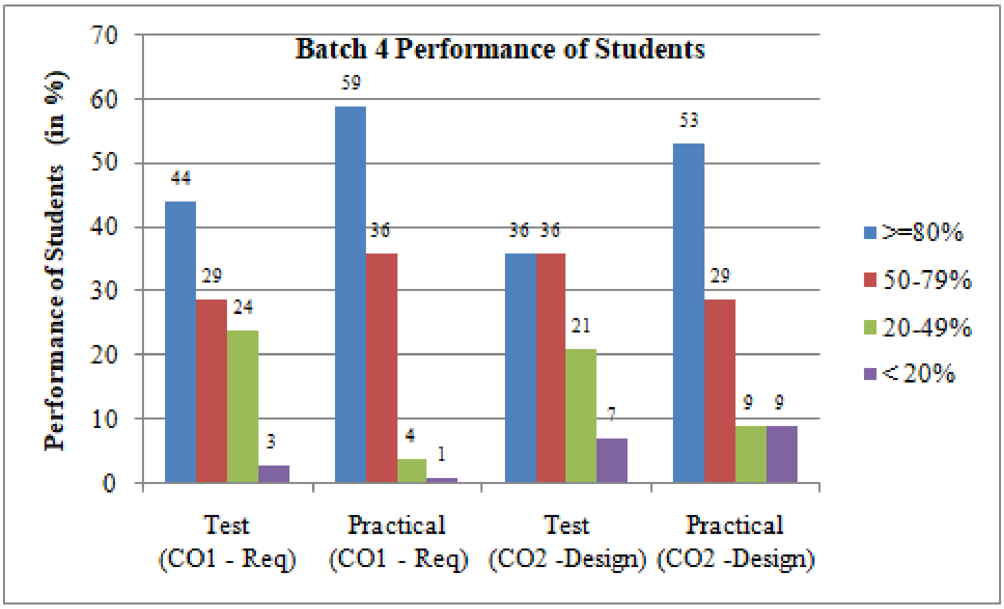
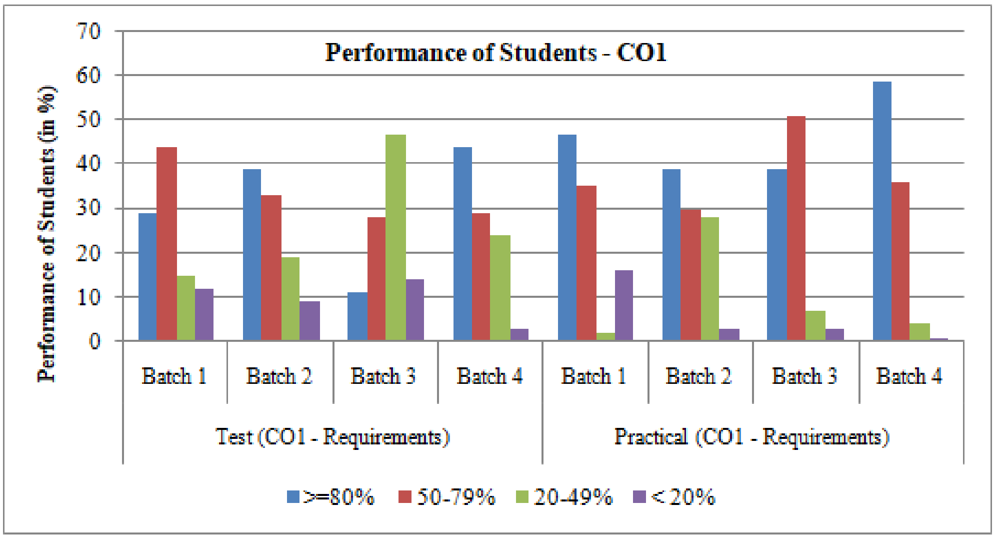


Figure 13. Course end survey – sample question Q3



is measured using these responses: weighted average method and all recorded responses were used to calculate the overall feedback score. Rating 4 takes the weight 1, rating 3 takes the weight 0.75, rating 2 takes the weight 0.5 and the rating 1 takes the weight 0.25.

Table 7 shows the weighted average responses for each course outcome of the experimental group. It is evident that the students enjoyed learning and recorded good feedback about the course. The blended mode of teaching learning mode supported both the teachers and the learners to get better satisfaction *(RQ2)*.

Figure 14.
Course end survey – sample question Q4

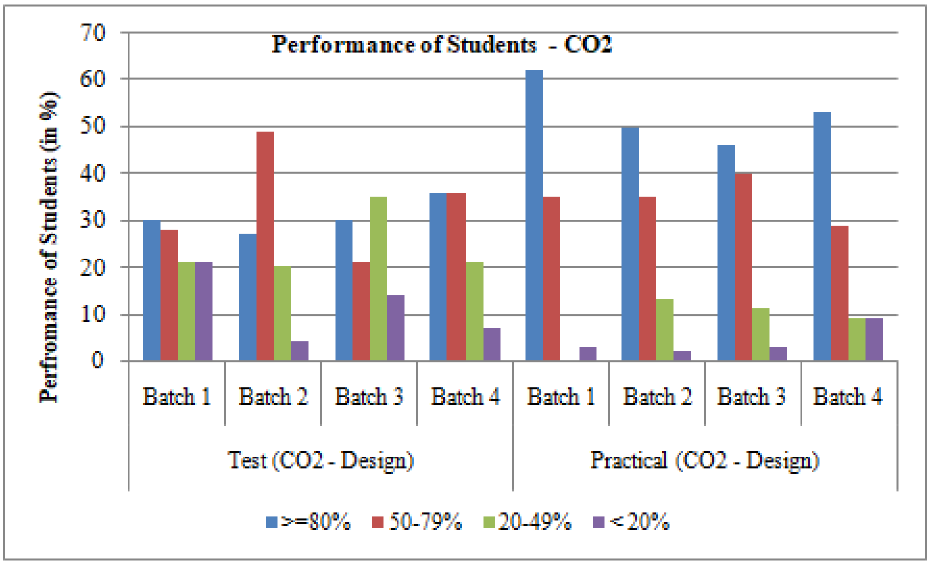
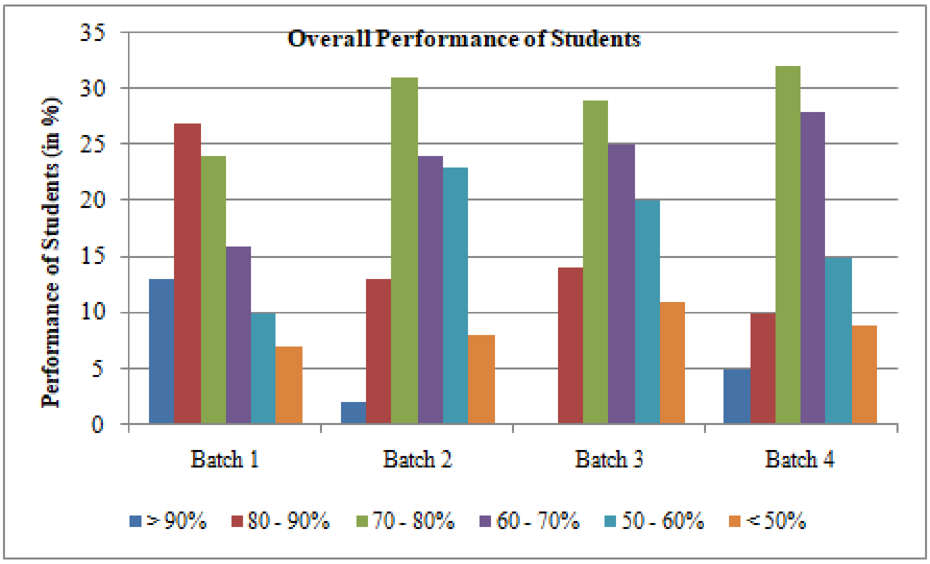


Figure 15.
Course end survey – text feedback Q5



CONCLUSION

Software Engineering course is the complex and challenging course for Engineering Course Faculty as it mostly involves theoretical concepts. The Course Instructor customizes the course plan using the blended learning approach so as to make the students to understand the Software Engineering principles practices. This blended learning approach help to fill the gap between the academic world and the industry by enabling the Graduates to adopt Corporate practices easily and efficiently. This

Table 7.
Students feedback on learning outcomes

Batch	Weighted Average of Responses			
	CO1	CO2	CO3	CO4
Batch1	79	79	77	72
Batch2	84	84	81	81
Batch3	83	84	80	84
Batch4	84	86	83	85

teaching learning approach is an enabler for attaining Student Learning Outcomes, promoting student performance thereby achieving Programme Outcomes (POs) also. Thus, the blended learning approach helps in setting up the effective learning environment for teaching the Software Engineering course in Undergraduate Engineering curriculum.

ACKNOWLEDGMENT

We acknowledge our gratitude to our Institute for supporting the practice of Active Learning Strategies and Blended Learning practices for this course.

CONFLICT OF INTEREST

The authors are not disclosing any conflict of interest. The authors themselves are responsible for the quality of this article and its composition.

SOURCE OF FUNDING

No financial aid or funding has been received from any source for this research.

REFERENCES

Alabbadi, A., & Qureshi, R. J. (2016). The proposed methods to improve teaching of software engineering. *International Journal of Modern Education and Computer Science*, (7), 13–21. doi:10.5815/ijmecs.2016.07.02

Alramouni, S., & Alkhawar, H. (2018). Strategies to improve at-risk students' learning and achievement: Undergraduate software engineering course case study. Springer. doi:10.1007/978-3-319-60011-6_3

Bano, M., Zowghi, D., Kearney, M., Schuck, S., & Aubusson, P. (2018). Mobile learning for science and mathematics school education: A systematic review of empirical evidence. *Computers & Education*, *121*, 30–58. doi:10.1016/j.compedu.2018.02.006

Chen, J., Lu, H., An, L., & Zhou, Y. (2009). Exploring teaching methods in software engineering education. In *Proceedings of 4th International Conference on Computer Science and Education*. Nanning.

Cummings, C., Mason, D., Shelton, K., & Baur, K. (2017). *Active learning strategies for online and blended learning environments*. IGI Global. doi:10.4018/978-1-5225-1803-7.ch006

Eison, J. (2010). Using active learning instructional strategies to create excitement and enhance learning. [Active Learning]. *Jurnal Pendidikantentang Strategi Pembelajaran Aktif*, *2*(1), 1–10.

Fonseca, V. M., & Gómez, J. (2017). Applying active methodologies for teaching software engineering in computer engineering. In *Proceedings of IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, 12*(3), 147-155. IEEE. doi:10.1109/RITA.2017.2738178

Huan, S. (2012). The innovation of software engineering course in the non-computer professional. *Procedia Environmental Sciences*, *12*(12), 1253–1256. doi:10.1016/j.proenv.2012.01.417

Hubscher, T., & Narayanan, H. (2003). Constructive and collaborative learning of algorithms. In *Proceedings of the 344th SIGCSE Technical Symposium on Computer Science Education*, (pp.6-10). ACM. doi:10.1145/611892.611919

Liu, X., Wang, X., & Wang, R. (2013). Application of blended learning in data structures and algorithms course teaching. In *Proceedings of International Conference on Education Technology and Information System ICETIS.* Atlantis Press. doi:10.2991/icetis-13.2013.245

Maguire, P., Maguire, R., Hyland, P. & Marshall, P. (2014). Enhancing collaborative learning using pair programming: Who benefits. *All Ireland Journal of Teaching and Learning in Higher Education (AISHE-J), 6*(2), 1411-1425.

Malmi, L., & Korhonen, A. (2008). Active learning and examination method for data structures and algorithms course. *Teaching of Programming, LNCS*, *4821*, 210–227. doi:10.1007/978-3-540-77934-6_17

Nail, B., & Ammar, W. A. (2017). Mobile learning education has become more accessible. *American Journal of Computer Science and Information Technology*, *5*(2).

Pandey, J. (2012). Use of e-learning in Uttarakhand school education system: case study of open source e-learning tools for fundamental mathematics and sciences. *International Journal of Computer Science and Technology*, *3*, 80–82.

Ramachandran, M. (2017). Technology enhanced active learning in software engineering. In *Proceedings of 9th International Conference on Computer Support Education CSEDU 2017* (pp. 242–248). Paula Escudeiro. SCITEPRESS – Science and Technology Publications. doi:10.5220/0006257602420248

Ramos, C. S., Kosloski, R. A. D., & Venson, E. (2018). TBL as an active learning-teaching methodology for software engineering courses. In *Proceedings of the XXXII Brazilian Symposium on Software Engineering SBES '18* (pp. 289-297). ACM NewYork. doi:10.1145/3266237.3266253

Venson, E., Figueiredo, R., Silva, W., & Júnior, L. C. M. R. (2016). Academy-industry collaboration and the effects of the involvement of undergraduate students in real world activities. In *Proceedings of IEEE Frontiers in Education Conference FIE 2016* (pp. 1-8). IEEE. doi:10.1109/FIE.2016.7757394

Yang, Y. F., Chien, L. C., & Chih, C. K. (2016). Learning motivation and retention effects of pair programming in data structures courses. *Journal of Education*, *32*(3), 249–267.

Yessenova, K., Parker, J., Sadvakasova, Z., Syrgakbaeva, A., & Tazhina, G. (2020). Kazakhstani e-learning practice in higher education: The key trends and challenges. *International Journal of Adult Education and Technology*, *11*(1), 1–21. doi:10.4018/IJAET.2020010102

A. M. Abirami is a Computer Science Engineer from Bharathiyar University, Coimbatore where she awarded the B.E. degree with first class distinction in 1999. She then served as a Software Engineer in Tata Consultancy Services Ltd. from 1999 – 2006 where she worked as a QA member. She did her M.E. degree from Anna University Tirunelveli with first class distinction in the year 2010. Since 2010, she is working as Assistant Professor in Thiagarajar College of Engineering, Madurai. She received her PhD Degree in the area of Text Analytics and Semantic Web from Anna University Chennai in 2018. She is interested in improving Teaching Learning methodologies of Engineering Education. She has earned "Cambridge International Certificate for Teachers and Trainers", trained by Wipro's Mission10x programme. She has been one of the top performers in the ICT Tools for Blended and Online learning FDP conducted by IIT Bombay. She has implemented many Active Learning Strategies for the courses Data Structures, Programming and Software Engineering to UG students. She is the professional member of ACM professional society.

S. Pudumalar is working as anAssistant Professor in the Department of Information Technology at Thiagarajar College of Engineering, Madurai.She received her UG and PG degrees with the specialization of Computer Science and Engineering. She has a passion for teaching as well as Research. She is an active member of pedagogical activities including Curriculum Design and Teaching Learning Process.Her recent publication on Engineering Education includes Effectiveness of Mobile Learning at TCE, India: A Learner Perspective in IEEE International ConferenceT4E2015. Her current research interests include predicting the chances of offering scholarship to students.

S. Thiruchadai Pandeeswari, holds a Post graduate Engineering degree in Multimedia Technology and is currently pursing her Doctoral research in the area of service computing. Her research interests include Service Computing, Software Defined Networking and Engineering Education Pedagogy.