Journal of Information Technology Education: Innovations in Practice

An Official Publication of the Informing Science Institute
InformingScience.org

JITEiip.org

Volume 17, 2018

ACTIVITY ORIENTED TEACHING STRATEGY FOR SOFTWARE ENGINEERING COURSE: AN EXPERIENCE REPORT

Jeevamol Joy*	Cochin University of Science and Technology, Kochi, India	jeeva.loy@gmail.com
Renumol V G	Cochin University of Science and Technology, Kochi, India	renumolvg@gmail.com

* Corresponding author

ABSTRACT

Aim/Purpose	This paper presents the findings of an Activity-Oriented Teaching Strategy (AOTS) conducted for a postgraduate level Software Engineering (SE) course with the aim of imparting meaningful software development experience for the students. The research question is framed as whether the activity-oriented teaching strategy helps students to acquire practical knowledge of Software Engineering and thus bridge the gap between academia and software industry.
Background	Software Engineering Education (SEE) in India is mainly focused on teaching theoretical concepts rather than emphasizing on practical knowledge in software development process. It has been noticed that many students of CS/IT background are struggling when they start their career in the software industry due to inadequate familiarity with the software development process. In the current context of SE education, there is a knowledge gap between the theory learned in the classroom and the actual requirement demanded by the software industry.
Methodology	The methodology opted for in this study was action research since the teachers are trying to solve the practical problems and deficiencies encountered while teaching SE. There are four pedagogies in AOTS for fulfilling the requirements of the desired teaching strategy. They are flipped classroom, project role-play for developing project artifacts, teaching by example, and student seminars. The study was conducted among a set of Postgraduate students of the Software Engineering programme at Cochin University of Science and Technology, India.

Accepted by Editor Janice Whatley | Received: April 20, 2018 | Revised: August 18, August 30, September 4, 2018 | Accepted: September 5, 2018.

Cite as: Joy, J., & V G, R. (2018). Activity oriented teaching strategy for software engineering course: An experience report. *Journal of Information Technology Education: Innovations in Practice, 17,* 181-200. https://doi.org/10.28945/4116

Contribution	AOTS can fulfil both academic and industrial requirements by actively engaging the students in the learning process and thus helping them develop their professional skills.
Findings	AOTS can be molded as a promising teaching strategy for learning Software Engineering. It focuses on the essential skill sets demanded by the software industry such as communication, problem-solving, teamwork, and understanding of the software development processes.
Impact on Society	Activity-oriented teaching strategies can fulfil both academic and industrial requirements by actively engaging the students in the SE learning process and thus helping them in developing their professional skills.
Future Research	AOTS can be refined by adding/modifying pedagogies and including different features like an online evaluation system, virtual classroom etc.
Keywords	software engineering education, activity oriented teaching, learning environment, flipped classroom

INTRODUCTION

Current Software Engineering (SE) education is mostly based on a classroom learning model that does not meet the essential requirements of a subject that demands practical and interdisciplinary approaches for solving problems (Shaw, 2005; Varma & Garg, 2005). For achieving the desired learning objectives of SE, various learning environments are being tried by software engineering educators worldwide. Even then, additional needs exist to improve the learning process and the overall quality of a learning environment (Garg & Varma, 2015). According to Garg and Varma (2015), the requirements of an effective and sustainable SE learning environment are broadly classified into climatic (authentic learning, self-learning, learning from failures and success, motivate students, etc.) and systemic (scalability, portability, analyzability, reusability, and fault tolerance). The current software engineering courses taught at the undergraduate/postgraduate level in India focus more on software processes and management of these processes rather than software design and design qualities (Reddy & Nori, 2014). While designing software engineering courses, it is essential to consider what knowledge is important for a software professional (Lethbridge, 2000).

Software engineering graduates, when beginning their careers in the software industry, do not always possess the necessary skills, abilities, or knowledge that are demanded by the industry (Radermacher, Walia, & Knudson, 2014). According to the recruiters, newly hired graduates are mainly lacking in the areas of project experience, oral communication, written communication, problem-solving, software tools, teamwork, and working with customers. The deficiency in these skill sets limits the productivity of newly hired software engineering graduates. The repetitive complaint from the software industry is that computer science and software engineering graduates are not well trained during their undergraduate/postgraduate studies for their future careers in the software industry (Begel & Simon, 2008). This lack of student preparation is not limited just to programming skills or other computer science concepts, but also includes the understanding of software engineering concepts such as software development processes. In some cases, there is a large requirement gap between students' skills and the expectations of industry managers or hiring personnel, which prevent students from getting job offers and meeting the expectation level of the software industry (Miller & Dettori, 2008).

In the current context of SE education, there is a knowledge gap between the theory learned in the classroom and the actual requirements demanded by industry. This motivated the authors to design and develop a better teaching strategy which may bridge the required knowledge gap between academia and industry. This paper describes the experience of a teacher; using an Activity-Oriented Teaching Strategy (AOTS), which was adopted for a Software Engineering course at the postgraduate level in Cochin University of Science and Technology, India. The objective of the study is to develop

an effective teaching strategy for the Software Engineering course by which students can acquire more practical knowledge of SE and get a better understanding of the applicability of theoretical concepts. The methodology opted for this study was action research since the teachers are trying to solve the practical problems and deficiencies encountered while teaching SE. Action research is usually undertaken by practitioners to improve their teaching practices (Corey, 1954). Here the focus of our research is to develop a teaching strategy for SE, which helps the students acquire practical knowledge of software engineering so that they can successfully operate in an industrial environment.

Different learner-centered pedagogies have evolved in Engineering Education which motivate and empower students by giving them control over their learning. The flipped classroom pedagogy is one of the most popular among these nowadays and is included in AOTS. In a flipped classroom, the information-transmission component of a traditional face-to-face lecture is moved out of class time. In this pedagogy, active and collaborative tasks are included in the class hours (Abeysekera & Dawson, 2015). In the flipped classroom model, students prepare themselves for the lesson by watching videos, listening to podcasts, and reading articles. Students use this knowledge while active learning activities are conducted in the classroom with the guidance of a teacher. According to Hamdan, McKnight, McKnight, and Arfstrom (2015), the flipped classroom is not a defined model, instead it is a model that teachers use for supplementing the demands of students by using different types of tools. Since educators in different countries use flipped classroom with various methods, the flipped classroom concept was changed to a flipped classroom approach.

AOTS is a student-centric learning approach in which learners play a central role and take responsibility for their own learning. In this kind of learning, the instructor acts as a facilitator for encouraging learning and being a guide in the whole learning process. The primary goal of this teaching strategy is to prepare students to understand the concepts of the subject and to apply the acquired theoretical knowledge to practical cases of software projects, preparing them for the software industry by mainly focusing on the skill sets which are lacking in conventional SE courses. Hence for the study, the Research Question (RQ) is framed as follows:

> RQ: Whether the Activity-Oriented Teaching Strategy (AOTS) helps students to acquire practical knowledge of Software Engineering and thus bridge the gap between academia and the software industry?

The rest of the paper is organized as follows. A review of the literature is included, which mainly focuses on studies conducted at different countries in the area of software engineering education followed by the expected learning outcomes from a postgraduate level SE course. The design of AOTS, procedures, and detailed execution of each teaching method in AOTS are explained subsequently. The evaluation and results sections provide detailed findings of the study as well as the student feedback about AOTS. The paper is concluded with the outcomes of the study, limitations, and future works planned.

RELATED WORK

There are several guidelines available for the basic and core courses of Software Engineering Education (SEE). In addition to these, the other dimensions of SEE that should be considered are interdisciplinary skills, practice experience, communication skills, skills for continuing education, and professionalism (Mishra, Ercil Cagiltay, & Kilic, 2007). A study was conducted to observe how these dimensions are handled in SE courses at different universities in the world; it has been observed that the distribution of courses in terms of these dimensions vary from university to university and are missing in some universities. In India, almost all major universities offer undergraduate programs in Computer Science and a few offer postgraduate programs as well. But very few offer courses or degree programs with a major in Software Engineering (Garg & Varma, 2008). Usually, SE is offered as just one of the subjects in a Computer Science program. Thus most Computer Science graduates

study SE at the most for only one semester and, for some students, this is the only opportunity to get familiarized with SE before starting their career as Software Engineers. Hence it becomes very important that this course imparts the knowledge and skills which are expected by the software industry.

Typically, a Software Engineering course aims to impart the learning outcomes such as technical capability, teamwork, problem-solving, communication/presentation skills, technical documentation, and software engineering practices. Software engineering (SE) teachers around the world are striving to make learning situations that can accomplish their expected learning outcomes (Garg & Varma, 2015). A study conducted by Fonseca and Gómez (2017) at two different universities in Chile detailed the development of student projects in the area of software using the framework of active methodologies. The pedagogies included in their framework were problem-based learning and agile software engineering, which enabled the students to acquire a deeper knowledge and apply it in a practical way according to a work plan. In this framework, students were assigned different project roles in real projects (project leader, programmer, tester, etc.) and were able to work in a dynamic environment using problem-based learning. Even though this learning framework gives importance to develop the skills of autonomous learning, creative product development, and teamwork, this method is lacking in developing communication/presentation skills, technical documentation, and software engineering practices.

In another study (Paez, 2017) conducted at Universidad Nacional de Tres de Febrero, Argentina, a flipped classroom approach combined with other non-traditional teaching techniques was adopted to teach a software engineering course. For conducting the experiment with the flipped classroom, the instructors created relevant course materials and planned in-class and out-of-class activities with the help of a virtual classroom. The virtual classroom was mainly used for file sharing and to extend the interaction between instructors and students beyond the in-class time. The additional teaching strategies involved in this approach are continuous practice, teaching by example, and use of real-world tools. The main pedagogy used in this study is a flipped classroom, and it is found to be effective for teaching SE. The methodology used in this study does not include pedagogies that develop other required skills like teamwork, technical documentation, and software engineering practices that include processes and quality assurance techniques.

A Case-Based learning model (COSSEEd) (Garg, Sureka, & Varma, 2015) with well-designed SE case studies as primary learning objects has been used in a study conducted at IIIT-Hyderabad, India. COSSEEd provides opportunities for students to solve the challenges embedded in a context (case-study) collaboratively and, thereby, engage in authentic activities similar to that of a software engineer. The experimental analysis reveals that the model successfully achieves the cognitive goals of a typical SE course. Other than this, COSSEEd also supports teamwork, problem-solving, technical competence and, communication skills, but is mainly lacking in technical documentation and other SE practices. Another educational framework (Yadav & Xiahou, 2010) that focused on integrated project-based learning was experimented with at Xiamen University, China to effectively develop crucial software engineering skills. According to the authors, an integrated project-based learning approach can be used for all the projects carried out in the software engineering course. The main focus of this model was requirements engineering, software design, implementation, teamwork, project management, and project documentation. In this framework, instructors used the pedagogy of project-based learning, which gives importance to teamwork, problem-solving, and technical documentation, but this framework was also lacking in areas like technical competence and communication/presentation skills.

There is an evident change in the methodologies to teach software engineering along with the growth of technology. Some examples are game-based teaching and learning platforms (Pieper, Lueth, Goedicke, & Forbrig, 2017; Tillmann, De Halleux, & Xie, 2011), cloud-based learning environments (Rana, Saleh, & Ghazali, 2017), and sensor-based cognitive approach (Gandhi, 2016), which are guid-

ed by technology rather than classroom pedagogies and therefore beyond the scope of the current study.

From the existing literature, it was found that different methodologies and pedagogies were experimented with to develop effective teaching strategies for Software Engineering courses. The major pedagogies implemented in the existing studies are project-based learning, problem-based learning, context-based learning, collaborative learning, active learning, and flipped classroom. Most of the methodologies resulted in visible improvement of learning outcomes. However, it is difficult to achieve all the learning outcomes of an SE course with any one methodology. We have analyzed the learning outcomes of each existing SE learning model and found that each model had its share of advantages as well as limitations in imparting different essential skills. The activity-oriented teaching strategy (AOTS) combines different pedagogies to impart all the essential skills required of a software engineering course. The learning outcomes of AOTS and existing SE learning models mentioned in the literature are compared in Table 1.

Table 1: Learning outcomes of AOTS and existing SE learning models

Learning Outcomes	**Description**	Active Methodologies (Fonseca et al., 2017)	Flipped classroom (Paez, 2017)	COSSEEd (Garg & Varma, 2015)	Integrated Project- based learning (Yadav & Xiahou, 2010)	**AOTS**
Technical Competence	An ability to apply knowledge of math, science, and software engineering as well as collect, analyze, and interpret data.	✓	✓	✓	×	✓
Team work	An ability to function on multidisciplinary teams.	✓	×	✓	✓	✓
Problem Solving	An ability to identify, formulate, and solve software engineering problems using a well-defined engineering process.	✓	✓	✓	✓	✓
Communication/ Presentation skills	An ability to communicate/present effectively.	✓	×	✓	×	✓
Technical documentations	Prepare project documents involved in a software development project.	×	×	×	✓	✓
Software Engineering Practices	Acquire knowledge about real software project scenarios in industry related to process/product, design, quality assurance, etc.	×	✓	×	×	✓

The experiment with AOTS was conducted for a set of postgraduate students; hence, it is worthwhile to discuss the learning outcomes required for a postgraduate level course in SE. Bloom's taxonomy outlines a hierarchy of cognitive-learning levels ranging from knowledge of specific facts and conventions to more advanced levels of comprehension, application, analysis, synthesis, and evalua-

tion (Azuma, Coallier, & Garbajosa, 2003). Students of postgraduate level SE courses are expected to attain up to the fifth level (synthesis) of Bloom's taxonomy. The Curriculum Guidelines for Graduate Degree Programs in Software Engineering (The Graduate Software Engineering 2009-GSwE2009) is a set of recommendations for a master's level graduate program in Software Engineering (Adcock et al., 2009). According to GSwE2009, the important recommendations for graduates of a master's SE program are:

1. Master the Core Body of Knowledge (CBOK)
2. Master software engineering in at least one application domain (e.g. finance, medical, retail, etc.)
3. Master at least one Knowledge Area (KA) from the CBOK to at least the Bloom Synthesis level.
4. Be able to make ethical professional decisions and practice ethical professional behavior.
5. Be an effective member of a team, including teams that are international and geographically distributed, effectively communicate both orally and in writing, and lead in one area of project development, such as project management, requirements analysis, architecture, construction, or quality assurance.
6. Understand and appreciate feasibility analysis, negotiation, and good communications with stakeholders in a typical software development environment, and be able to perform those tasks well; have effective work habits and be a leader.
7. Be able to learn new models, techniques, and technologies as they emerge, and appreciate the necessity of such continuing professional development.

While designing AOTS, the instructors considered both the advantages and limitations of existing pedagogies for teaching SE and the learning outcomes demanded by the course. The AOTS was designed to bridge the gap between the learning outcomes currently accomplished and the learning outcomes demanded by an SE course. The details of different teaching methods applied in AOTS for teaching SE are detailed in the next section.

DESIGN OF THE AOTS EXPERIMENT

Typically, a Software Engineering course aims to impart learning outcomes such as technical capability, teamwork, problem-solving, communication/presentation skills, technical documentation, and software engineering practices. All these essential skills cannot be acquired by the traditional lecture-based method of teaching. The major problem with the lecture-based approach is that the students are sometimes a passive audience during lectures and do not get actively involved in the learning process. This passive mode of learning is a great worry for academia as well as software industry, as it does not mold the students for creative product development and research (Yadav & Xiahou, 2010).

For AOTS, we have organized the syllabus of the SE course and grouped topics according to their nature. The existing teaching strategies in the literature were also analyzed and each group of topics correlated with the best-suited teaching strategy. For example, the method of developing project artifacts was applied to teaching software life cycles and models, whereas the flipped classroom approach was applied to product metrics, project metrics, project cost estimation, and project scheduling for getting better learning outcomes. AOTS combines different pedagogies like the flipped classroom, teaching by example, project role-plays for artifacts development, and student seminars. The pedagogies applied under AOTS are shown in Figure 1.

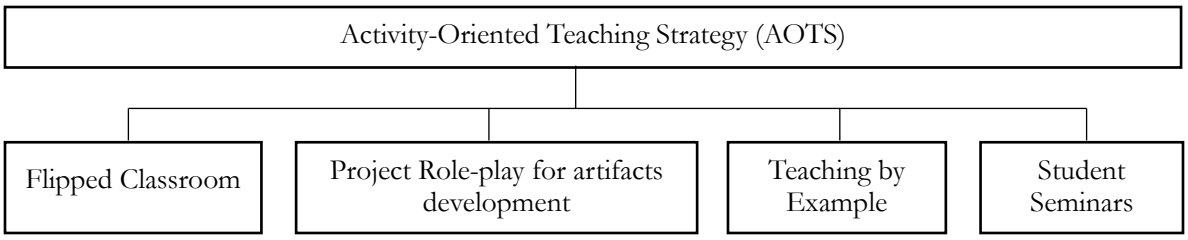

Figure 1: Pedagogies included under AOTS

The experiment was conducted for a group of 15 students who had registered for a master's (Software Systems) program at Cochin University of Science and Technology, India. Fourteen students had completed their Bachelor of Technology (B.Tech) program in the Computer Science /Information Technology stream and one student had completed her B.Tech. program in the Electronics and Communication Engineering stream. The age group of participants was between 22 and 27. Out of the 15 students, three students directly joined the program after completing their graduate-level course. Two of them had industrial experience of 2 to 3 years. Others took a 2 to 4 year break after their graduate-level course and then joined this post-graduate program. The duration of the SE course was 12-weeks with a 4-hour teaching session each week. The course was managed by one teacher and two technical assistants. The masters program was started by the university in the year 2013. From 2013 onwards, the course had been taught in a traditional lecture-based approach. The instructors noticed that students were struggling while doing their academic projects. By considering the essential requirements of SE that demands practical knowledge and problem-solving skills, we decided to conduct Activity-Oriented Teaching Strategy in the year 2017. At the end of the course, the learning outcomes of AOTS were measured by evaluating the project artifacts developed by the students, assessment of in-class activities, and by student opinion survey. The student opinion survey was conducted with a set of open-ended questions related to the teaching methods applied under AOTS (See the Appendix for the full survey). For each of the survey question, students were asked to write the benefits/outputs or the hindering factors of each teaching method in AOTS. Students' opinions about the method are discussed in detail under evaluation and results section. The details of AOTS methods applied in the classroom are elaborated in the following sections.

Flipped Classroom

We decided to try the flipped classroom approach under AOTS, which proposes a reversal of the traditional class flow (Bergmann & Sams, 2012). In the typical class flow, the "in-class" time is dedicated to lecture where the teachers explain the concepts and then distribute some work/exercises, which students have to complete at home. In the flipped classroom approach, students carry out the lecture at home, for example watching a video or reading an article and then the "in-class" time is used for interactive activities such as problem-solving and discussions (Gannod, Burge, & Helmick, 2008). The Flipped Classroom Model (FCM) has received great attention due to the benefits attributed by it in wide range of subjects, like STEM and ICT (Bishop & Verleger, 2013). FCM significantly enhanced students' cognitive learning outcomes in the context of Computing Curricula (Horton, Craig, Campbell, Gries, & Zingaro, 2014; Reza & Baig, 2015). More specifically, FCM has been proven for its positive results in Software Engineering courses (Amresh, Carberry, & Femiani, 2013; Gannod et al., 2008).

While selecting the topics for conducting the flipped classroom approach, the instructors made sure that the knowledge of the selected topics are important for the software professional. The topics selected under this category are architectural design, product metrics, project metrics, project cost estimation, and project scheduling. Most of the topics selected under this teaching method have applicability in the software industry related to project design and management activities. The in-class activities were selected in such a way that they will be helpful for the students in acquiring practical knowledge in the selected areas. Appropriate course materials were chosen for each of the selected

topics and shared with the students 2-3 days prior to the start of the in-class activities. Course materials included mainly videos and textbook references. Google classroom was used for sharing the course materials with the students. The in-class activities were conducted based on the out-of-class activities carried out by the students prior to the in-class time. The instructors clarified student doubts during the in-class time and gave required guidance for getting the expected learning outcomes from the students. The details of activities planned for the flipped classroom approach are given in Table 2.

Table 2: Details of in-class and out-of-class activities

Activity No	Topic	Out-of-class activity/material	In-class activity
1	Architectural design	Video lecture	Draw the architectural diagram of the student projects(Undergraduate level)
2	Process / project metrics	Text book reference, Video lecture	Problem solving and discussion -Defect removal efficiency, Integrity of software and customization index.
3	Product Metrics	Video lecture	Function point estimation for a software project
4	Project estimation	Go through one or more online model for project estimation from web based courses.	Discussion on Project estimation factors, Time estimation for students' Undergraduate project.
5	Project scheduling	Text book reference, Video	Problem solving – Earned value analysis

When the flipped classroom approach is considered, success of the method mostly depends on the completion of out-of-class activities on time and the quality/appropriateness of materials (videos, textbook, references) given for out-of-class activities (O'Flaherty & Phillips, 2015). Therefore, students were asked to respond to the following questions related to the flipped classroom approach at the end of the course.

Have you completed your outclass activities on time?

Whether the learning materials shared for out-of-class learning was appropriate and helpful in completing the in-class activities?

Along with this, the instructors evaluated all the documents submitted by the students as part of each in-class activity and gave feedback to each of them. The evaluation details of in-class activities and student opinions are included under the evaluation and results section.

TEACHING BY EXAMPLE

The use of concrete examples while teaching is an easy way to capture student attention and make the concept simple for them to understand and remember (Kember & Kwan, 2002). In our experiment using AOTS, the method of teaching by example is mainly used by the instructors to teach different verification and validation methods, which comes under the topic "quality assurance of software." One of the instructors had eight years of industrial experience in software development, testing, and analysis, which helped her in using real examples and scenarios associated with project development in an enriching way.

This method of teaching was mainly adopted for the better understanding of different quality assurance techniques. The instructor used real project examples in the classroom to understand the different testing scenarios that come with the quality assurance of software. The instructor had highlighted the various modes of testing, such as unit testing, integration testing, system testing and regression testing, that are done in the software industry, pointing out which technique should be adopted in a particular scenario for ensuring optimum test results. For example*, if there is a need of adding a new column to an existing table in the database (part of a running project), regression testing should be done after making the table change in order to make sure that, the table change doesn't affect the other part of the application in any way.* The instructor presented different testing scenarios such as changing an input file by adding more details in it, and most of the students could correctly identify the required testing strategies (unit testing followed by integration testing) for the given scenario. This method of teaching by example has given first-hand knowledge of quality assurance techniques in a typical industrial environment.

The example-based teaching method influences students' attitudes and interests towards the subject and leads to greater learning achievements (Tai, Leou, & Hung, 2015). But the learning achievement depends on the selection of an appropriate example. While choosing examples, the pedagogical reasoning from the perspective of learner and teacher creates two interlinked problems (Shafto, Goodman, & Griffiths, 2014). For the teacher, the problem is to choose suitable examples that will help the learner to infer the correct concept easily. For the learner, the problem is to infer the same concept, conveyed by the teacher through these examples. While adopting examples-based teaching method in AOTS, the instructors gave importance to the above factors to achieve maximum learning outcomes.

The question included in the opinion survey to get the feedback from the students about teaching by example method is

> "*Whether the examples used by the instructor in the classroom were appropriate and helpful to understand the concepts clearly?"*

The student opinions are detailed under the evaluation criteria of student opinion survey.

Project Role-play for Artifacts Development

The main purpose of an activity-oriented teaching strategy in the software engineering course is to gain some software engineering experience which cannot be obtained by traditional lecturing. Project role-play for developing project artifacts was introduced in AOTS since all the software projects involve the creation of artifacts such as Software Requirement Specification (SRS), Design document and Test plan documents. The instructors think this method will help the students in the future to do their academic projects in an authentic way and also to bridge the gap of industrial requirements when they enter into the software industry.

In the planning phase of the study, the instructors decided on the project artifacts that the students have to develop during the course. When the lifecycle of a software development project is considered, the most prevalent project artifacts are SRS, design document, and test plan. Therefore these artifacts were selected to be developed by the students. Meanwhile, students were divided into groups of 3 or 4, and they were asked to revise the major projects they did at the undergraduate (B.Tech.) level. According to the planned strategy, students were asked to work as a team to collect the requirements and to prepare the SRS, design, and test plan documents. The instructors acted as facilitators during group activities and guided them on how to collect requirements and what things should be taken care of to get maximum benefits from the group activity. Each student in the group played the role of a client for their own undergraduate project and the rest of the students in the group were asked to play the roles of System Analyst, Designer, and Tester. For a group of 4 members, the role play among the group is as shown in Figure 2. In this way every student in the group got an opportunity to get familiarized with the creation of all planned project documents and also got a hands-on experience in software development.

The development of project artifacts started with a requirements analysis phase. Requirements analysis and gathering is one of the most important tasks in the software development phase (Hofmann & Lehner, 2001). The student playing the role of system analyst needed to collect all the requirements from the client person and to prepare the SRS document based on the requirements. The SRS created by the analyst is used as the base document to prepare the design and test plan by the designer and tester members in the student group. The discussions among the students within the team led to developing good quality artifacts, due to continuous interaction among the team.

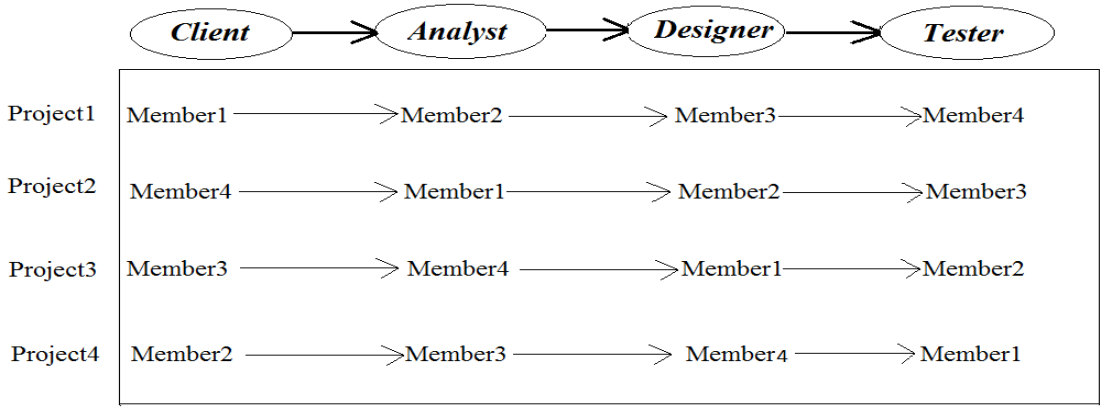

Figure 2: Project role play during project artifacts creation

In the student opinion survey, students were asked about the feedback of project artifacts creation, which is a project-supported teaching method.

> *Do you think the project artifacts (SRS, Design Document, and Test Plan) created helped in enhancing your practical knowledge?*
>
> *Whether the activities behind the creation of project artifacts improved your skills to work in a project context within a team?*

In the software engineering context, no two projects are exactly alike and the processes involved from requirements engineering to validation are different for each project (Gary, 2015). To professionally prepare students, experience in a project context is essential. In the current experiment, each student is involved in three or four projects to gain experience of working in different project contexts.

STUDENT SEMINARS

Teaching has proved to be an important opportunity for recognizing one's own ignorance and thereby rendering oneself open to the possibility of learning (Cortese, 2005). Presentation skills and interpersonal skills are crucial in the software development industry. These aspects were considered in AOTS. By including student seminars in this study, students had an opportunity to prepare and teach lessons, or parts of lessons, in front of the remaining students in the class, thereby improving their soft skills. Most of the students in the class had already undergone a basic course in software engineering at the undergraduate level. Since the students had basic knowledge about the subject, seminars were given for the topics that are the basics of software engineering, such as software life cycle models and software development principles. The instructors assigned one topic for each student to come prepared and teach in the class for one hour. The students prepared PowerPoint presentations and taught their assigned topic in the classroom. During student seminars, the instructors monitored the content of the presentations and gave constructive feedback to improve the overall content as well as their presentation skills.

Students consider personal transferable skills like presentation skills and communication skills most important for them in the future, and they require special training to enhance these skills (Haigh & Kilmartin, 1999).

In the student opinion survey, the question related to this teaching method was

> "*Do you think the individual presentations helped to improve your presentation and communication skills?*"

By including student seminars in AOTS, students were able to acquire these skills to a certain extent.

EVALUATION AND RESULTS

The AOTS was evaluated by three methods: by evaluating the project artifacts developed by the students, assessment of in-class activities, and by a student opinion survey.

BY EVALUATING PROJECT ARTIFACTS

During the first week of the course, the students were asked to write the high-level requirements of their own major projects at the undergraduate level. Requirements definition is a careful assessment of the needs that a software system is to fulfill (Ross & Schomn, 1977). It must state why a system is needed (context), it must state what system features will serve and satisfy this context (functional specification, stakeholders) and how the system is to be constructed (design constraints). Upon evaluating the high-level requirements document prepared by the students, the instructors found that students were not knowledgeable about how to write a requirements document and the details which should be included in the document. The documents prepared by the students contained only the objective of the projects and software modules they developed for their projects. As the course progressed, students worked in teams and submitted their Software Requirement Specification (SRS), design, and test plan documents based on the guidelines given by the instructors. These documents were evaluated by the instructors to analyze the extent to which AOTS can help students to improve their skills in requirements gathering, analysis, design, testing, and documentation. Upon evaluation of the documents, it was found that students acquired noticeable improvement in the construction of the project artifacts involved at different stages of project development. Students choose good document templates (IEEE template) for creating each of the artifacts that met the level of a software development industry standard. The SRS documents created by the students were of good quality and could be used by another person to create both design and test plans.

The students tried to incorporate as many details as possible of the functionalities in the design document including use-case and data flow diagrams. Some of the design documents contained architectural design and even small design considerations. Overall quality of the design documents was found to be above average. In classrooms, students created a unit test plan based on the project specification, and this was reviewed by a peer student, as practiced in the software industry, to capture the deficiencies in the test plan. By this, students improved in framing test plans covering the specifications in a comprehensive manner. Due to time constraints, the test plan document was created for only one function of the assigned project, and it was noted by the instructors that most of the students selected the least complex functionality of their respective project for test plan preparation. But now students had developed a basic understanding of how to write test cases, including the logical sequence of all the input data setup, input conditions, expected results, etc. Even though the content of all three documents were of good quality, noticeable documentation errors were found during the evaluation. The review comments for each document were shared with the students individually and students resubmitted the documents after making any necessary corrections.

In higher education, collaborative learning environments are helpful to achieve self-regulated learning and student motivation (Järvelä & Järvenoja, 2011). In AOTS, the creation of project artifacts was done by students in a collaborative environment under the guidance of an instructor. Students interacted within the team and started creating draft versions of the required documents. Later on, they

made different iterations in the documents and improved them to the desired level. In the student opinion survey, the respondents were asked: "*Whether the activities behind the creation of project artifacts enhanced your skills to work in a project context within a team?"* All the students reported that the method was helpful in understanding the project environment and in proceeding with project activities effectively. They also said that, as a result of the group activities, their interpersonal skills had also improved which is an essential skill required in the software industry.

Evaluation of In-Class Activities in the Flipped Classroom

The effectiveness of flipped classrooms is evaluated mainly by two methods. One is student feedback and the other method is to measure the student grades in the examinations conducted in between or at the end of the course (Karabulut-Ilgu, Jaramillo Cherrez, & Jahren, 2015). In the flipped classroom approach, we have collected the student feedback regarding the learning material sources shared with the students as part of out-of-class activities and the appropriateness and quality of in-class activities chosen. Also, the instructors evaluated the in-class activity of every student. Students had prepared the architectural diagrams of their undergraduate level projects for their in-class activity. During in-class time, students actively participated in discussions and interacted with the instructors to complete the architectural diagrams, and they were able to comprehend the system better by these activities. The instructors evaluated the architectural diagrams prepared during the in-class time, focusing on the structure of the software system which comprised the software elements, the relationships among them, and the properties of both elements and relations. Upon evaluating the architectural diagrams, the instructors noticed the increase in students' knowledge level which was lacking in the initial discussions on the topic. The instructors reviewed the student activity and emphasized the areas in which each student needed work. The in-class activity given for the topic, product metrics, was to determine the Function Point estimate of a software project. Students were able to estimate different performance indicators such as cost per unit of software delivered, staff resource per unit of software delivered, and elapsed time to deliver a unit of software. Also, students did the project estimation for their undergraduate level project. Students considered the essential resources that are required to complete the project. Size, effort, and cost were estimated based on the user requirements and data given based on past projects. Based on these data, students did the project scheduling. Different problems were given to do the Earned Value Analysis to find whether a project is running behind schedule, on schedule, or ahead of schedule. Overall, the in-class activities chosen were helpful for the students to improve their practical knowledge in the subject and exposure to the industrial working environment.

There is a need for balance between out-of-class preparatory activities and time spent with actual in-class activities (O'Flaherty & Phillips, 2015). A lack of engagement with the out-of-class activities results in variability of student preparedness, which is a learning challenge in the flipped classroom model. In our study, most of the students were able to complete the in-class activities within the allocated time. Some students faced difficulties and experienced time lag while doing in-class activities since they had not gone through their out-of-class activities before the in-class session. In such cases, students had to spend their in-class activity time to understand the concepts that they were supposed to complete before coming for the in-class time. The activities and percentage of students completed the in-class activities in the allotted time are given in Table 3.

Table 3: Details of percentage completion of in-class activities on time

No.	In-class activity	% of students completed the activity on time	Comments
1	Architectural design	73	The remaining % of students took extra time to complete their in-class activity.
2	Process/project metrics	85	
3	Product Metrics	100	
4	Project estimation	76	
5	Project scheduling	45	Students spend in-class time for out-of-class activities.

STUDENT OPINION SURVEY

An opinion survey was conducted to get the student feedback about the activity oriented teaching strategy conducted to teach the SE course at postgraduate level. Fourteen out of 15 students participated in the opinion survey. The feedback from the survey is presented below along with the student opinion, which is quoted in italic.

The survey respondents were asked the question *"Have you studied Software engineering/equivalent courses at undergraduate level? Yes /No. If 'yes', point out the factors that you have experienced with the new teaching strategy at the advanced level."* Students who had earlier undergone software engineering course in their undergraduate level said that the new teaching strategy was more concentrated on the practical aspects of SE and this activity-oriented methodology will be helpful for them to apply the learned concepts in real project scenarios when they enter the software industry.

> *"The new teaching strategy is focused on acquiring practical knowledge of SE and got a better understanding of the applicability of learned concepts"*
>
> *"Got an idea to handle practical problems in SE"*

When it comes to flipped classroom and project artifacts creation, both encompass the features of an active learning strategy which is a favored method in engineering education (Freeman et al., 2014). Active learning is particularly beneficial in small classes and found to be very effective for comprehending the concepts easily. Here the experiment was conducted for an engineering postgraduate level course consisting of 15 students. Active learning is generally defined as an instructional method that engages students in the learning process. In short, active learning requires students to do meaningful learning activities and think about what they are doing (Jensen, Kummer, & Godoy, 2015).

The benefits of flipped learning are flexibility, improved teacher-student interaction, and increased student engagement in the learning process (Karabulut-Ilgu et al., 2018). The students mentioned that they were able to grasp the content beyond memorization and basic knowledge. Here also, 12 out of 14 students said that the in-class activities were beneficial for them to clarify doubts during the in-class time. And two of them said they were not able to complete the out-of-class activities on time due to time factors.

> *"In-class activities were helpful to understand the concepts and clearing the doubts"*

The existing studies say active learning leads to improvement in examination performance that would raise average grades by half a letter, and when compared with traditional lecturing the failure rates of active learning have been reduced by 55% (Freeman et al., 2014). In the current study, 12 out of 14 students said that the in-class activities will be beneficial for them in the exam.

> *"All the out-of-class activities were not completed on time because of time issues and sometimes it affected the in-class exercises. The method was found to be very useful in problem-solving context and from exam point of view"*

According to the students' opinions the teaching strategy using project artifacts development was beneficial for all the students. The students' opinion is that this activity helped them to know how to (1) write the SRS, (2) prepare the design document, and (3) execute the test plan for different projects. They think this exercise will be helpful for them in the future to work as a team and improve their creative project development skills.

> *"Helped to understand how to make SRS, design and test plan for a new project"*

> *"Improved our team working skills and interpersonal skills"*

In the opinion survey, students were asked about the appropriateness of the examples used by the instructor in the class. By including concrete examples in our teaching and correlating these examples back to "real-world" systems and situations, students would be motivated towards the topic and understand the subject more effectively (Felder, Woods, Stice, & Rugarcia, 2000). In the current experiment, instructors tried to connect different testing methods with appropriate testing scenarios/cases in real software projects. Students also found that the teaching by example strategy was helpful for them to understand the real scenario, where different verification and validation methods need to be applied. All the students thought the examples used by the instructor were simple and relevant, and they were able to correlate between the theory and application.

> *"The simple and relevant examples used were very beneficial to understand the testing scenarios very well"*

Students considered soft skills like presentation skills and communication skills are most important for them in the future, and they required special training to enhance these skills. Thirteen out of 14 students opined that the seminars were helpful to enhance their confidence level and presentation skills.

> *"The individual topic presentations improved our confidence level and presentation skills"*

The overall opinion of the students about the new teaching strategy was that the method was concentrating on the practical side of the software engineering discipline. Three students opined that it was difficult to adapt to the new teaching strategy due to the time limitation of the course. Two students felt they could have spent more time on the in-class/out-of-class activities to get maximum benefit from the new teaching strategy. One student opined that from an exam point of view, traditional teaching is better when compared with the new activity oriented teaching strategy.

> *"New method would help the student to know more about the practical side, but it takes more time to complete all the activities"*

> *"Since the duration of the semester is limited, it was difficult to adopt the new teaching strategy. For the exam point of view, traditional teaching is more effective than new strategy since the new method is concentrating on the practical side of software engineering"*

Overall, AOTS emphasized the areas where traditional SE teaching methods are lacking. Students opined that AOTS helped them to acquire the practical knowledge of SE, to understand the applicability of the learned concepts, and to develop interpersonal/soft skills, thus helping the students for their professional careers.

CONCLUSION AND FUTURE WORK

This paper describes the findings of an activity-oriented teaching strategy (AOTS) which was adopted for a software engineering course at postgraduate level. It was found that AOTS helped students in acquiring practical knowledge of SE and how to apply the learned concepts on different project scenarios. The key features of AOTS are:

- Student centric.
- Promotes collaborative learning
- Uses active student learning.
- Provides many concrete, practical examples.
- Helps to meet industrial needs.

There are several complaints from the software industry sector that graduates are not well prepared for their professional careers and they are not ready to enter into the real work environment (Almi, Rahman, Purusothaman, & Sulaiman, 2011). The skills which are lacking are good communication skills, ability to follow processes and to be part of a team, and project management skills. This arises since the SE education environment differs from the Software Engineering industry environment. This experiment reveals that activity-oriented teaching strategies can fulfill both academic and industrial requirements by actively engaging the students in the learning process and thus helping them develop their professional skills.

Constructivism is a theory of learning which is very much applicable to software engineering education (Hadjerrouit, 2005a). According to this theory, knowledge must be actively constructed by learners, not passively transmitted by teachers. To get students more actively involved in knowledge construction, learner-centered pedagogies are essential (Hadjerrouit, 2005b). AOTS consists of different learner-centered pedagogies like flipped classroom, project role-plays for developing project artifacts, and student seminars. According to student feedback, the most useful pedagogy in AOTS was project role-play for developing project artifacts, which enabled them to acquire deeper knowledge about the creative software development procedure. Students opined that the flipped classroom approach was far more effective and engaging than the traditional lecture-based approach since this method gave more importance to problem-solving. In this method, students had to apply higher cognitive levels in order to analyze and evaluate solutions for project-based problems, which they have to deal with in an industrial atmosphere. In the teaching by example method, the students were able to correlate different quality assurance techniques with real project scenarios in the software industry. Student seminars had vastly improved confidence levels and presentation skills of the students.

Overall, AOTS emphasized the areas where traditional SE teaching methods are lacking. AOTS guided students towards acquiring theoretical knowledge as well as practical skill sets which are essential in the software industry and currently lacking in the newly recruited graduates. Thus, the research question has been answered as "Yes;" AOTS helps the students to acquire practical knowledge of Software Engineering and thus bridge the gap between academia and industry. In particular, students benefited from the activities such as project role-play, individual presentations, collaborative problem-solving tasks, and examples of quality assurance techniques in different project scenarios.

Beyond the positive results, students found difficulties in meeting the different milestones set by the course, due to time constraints. Also, AOTS could not encompass pedagogies for addressing managerial skills such as people management and conflict management due to time constraints, which is also an essential skill required in the software industry. The above problems could be eliminated through more efficient planning of the teaching strategy. At the initial stages of AOTS implementation, instructor load is comparatively high when compared to traditional teaching methods since precise planning and focused effort is required for each AOTS activity. Once the AOTS becomes refined based on the feedback and the learning outcomes, the instructor load will reduce considerably, since all the teaching aids for each activity will be in place before the commencement of the course.

According to the instructors, AOTS is a promising teaching strategy for software engineering courses. To make a generalization about the benefits of AOTS, further longitudinal cohort studies are required in this area to evaluate the learning outcomes. We are planning to continue this study in 2018 also, by refining AOTS based on our current experience, learning outcomes, and student feedback. In the present study, all the AOTS activities were evaluated manually. We are planning to include an online grading system for conducting and evaluating assignments for each activity. We are also plan-

ning to introduce virtual classrooms for improving the interactions among instructors and students beyond the regular class time so as to get maximum AOTS utilization and to improve the learning outcomes.

REFERENCES

Abeysekera, L., & Dawson, P. (2015). Motivation and cognitive load in the flipped classroom: Definition, rationale and a call for research. *Higher Education Research & Development, 34*(1), 1-14. https://doi.org/10.1080/07294360.2014.934336

Adcock, R., Alef, E., Amato, B., Ardis, M., Bernstein, L., Boehm, B., ... & Edson, R. (2009). *Curriculum guidelines for graduate degree programs in software engineering*. Integrated Software and Systems Engineering Curriculum Series. New York: NY: ACM,

Almi, N. E. A. M., Rahman, N. A., Purusothaman, D., & Sulaiman, S. (2011). Software engineering education: The gap between industry's requirements and graduates' readiness. In *Proceedings of IEEE Symposium on Computers & Informatics (ISCI),* 542-547. https://doi.org/10.1109/ISCI.2011.5958974

Amresh, A., Carberry, A. R., & Femiani, J. (2013). Evaluating the effectiveness of flipped classrooms for teaching CS1. In *Proceedings of the 2013 IEEE Frontiers in Education Conference*, 733-735. https://doi.org/10.1109/FIE.2013.6684923

Azuma, M., Coallier, F., & Garbajosa, J. (2003, September). How to apply the Bloom taxonomy to software engineering. In *Proceedings of 11th Annual International Workshop on Software Technology and Engineering Practice, IEEE*, 117-122.

Begel, A. & Simon, B. (2008). Struggles of new college graduates in their first software development job. *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, New York, NY, USA: ACM, 226–230. https://doi.org/10.1145/1352135.1352218

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

Bishop, J. L., & Verleger, M. A. (2013). The flipped classroom: A survey of the research. In *ASEE National Conference Proceedings,* Atlanta, GA, 30(9), 1-18.

Corey, S. M. (1954). Action research in education. *The Journal of Educational Research, 47*(5), 375-380.

Cortese, C. G. (2005). Learning through teaching. *Management Learning, 36*(1), 87-115. https://doi.org/10.1177/1350507605049905

Felder, R. M., Woods, D. R., Stice, J. E., & Rugarcia, A. (2000). The future of engineering education II. Teaching methods that work. *Chemical Engineering Education, 34*(1), 26-39.

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

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences*, 111(23), 8410-8415. https://doi.org/10.1073/pnas.1319030111

Gandhi, K. I. (2016). A teaching-learning model for software engineering courses through sensor-based cognitive approach. *International Journal of Engineering Education, 32*(2), 915-926.

Gannod, G. C., Burge, J. E., & Helmick, M. T. (2008). Using the inverted classroom to teach software engineering. In *Proceedings of the 30th International Conference on Software Engineering*, New York, NY: ACM, 777-786. https://doi.org/10.1145/1368088.1368198

Garg, K., Sureka, A., & Varma, V. (2015, December). A case study on teaching software engineering concepts using a case-based learning environment. In *1st International Workshop on Case Method for Computing Education (CMCE)*, 71-78. Retrieved from https://www.researchgate.net/publication/290084880_A_Case-Study_on_Teaching_Software_Engineering_Concepts_using_a_Case-Based_Learning_Environment

Garg, K., & Varma, V. (2008, April). Software engineering education in India: Issues and challenges. In *Proceedings of 21st Conference on Software Engineering Education and Training, CSEET'08*. IEEE, 110-117.

Garg, K., & Varma, V. (2015, February). Systemic requirements of a software engineering learning environment. In *Proceedings of the 8th India Software Engineering Conference*, ACM, 147-155.

Gary, K. (2015, September). Project-based learning. *Computer, 48*(9), 98-100. https://doi.org/10.1109/MC.2015.268

Hadjerrouit, S. (2005a). Constructivism as guiding philosophy for software engineering education. *ACM SIGCSE Bulletin, 37*(4), 45-49. https://doi.org/10.1145/1113847.1113875

Hadjerrouit, S. (2005b). Learner-centered web-based instruction in software engineering. *IEEE Transactions on Education, 48*(1), 99-104. https://doi.org/10.1109/TE.2004.832871

Haigh, M. J., & Kilmartin, M. P. (1999). Student perceptions of the development of personal transferable skills. *Journal of Geography in Higher Education, 23*(2), 195-206. https://doi.org/10.1080/03098269985461

Hamdan, N., McKnight, P., McKnight, K., & Arfstrom, K. (2015). *A review of flipped learning*. Retrieved from https://flippedlearning.org/wp-content/uploads/2016/07/LitReview_FlippedLearning.pdf

Hofmann, H. F., & Lehner, F. (2001). Requirements engineering as a success factor in software projects. *IEEE Software, 18*(4), 58. https://doi.org/10.1109/MS.2001.936219

Horton, D., Craig, M., Campbell, J., Gries, P., & Zingaro, D. (2014). Comparing outcomes in inverted and traditional CS1. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education,* New York, NY: ACM, 261-266.

Järvelä, S., & Järvenoja, H. (2011). Socially constructed self-regulated learning and motivation regulation in collaborative learning groups. *Teachers College Record, 113*(2), 350-374.

Jensen, J. L., Kummer, T. A., & Godoy, P. D. D. M. (2015). Improvements from a flipped classroom may simply be the fruits of active learning. *CBE—Life Sciences Education, 14*(1), ar5. https://doi.org/10.1187/cbe.14-08-0129

Karabulut-Ilgu, A., Jaramillo Cherrez, N., & Jahren, C. T. (2018). A systematic review of research on the flipped learning method in engineering education. *British Journal of Educational Technology, 49*(3), 398-411. https://doi.org/10.1111/bjet.12548

Kember, D., & Kwan, K. P. (2002). Lecturers' approaches to teaching and their relationship to conceptions of good teaching. In N. Hativa & P. Goodyear (Eds.), *Teacher thinking, beliefs and knowledge in higher education* (pp. 219-239). Springer, Dordrecht. https://doi.org/10.1007/978-94-010-0593-7_10

Lethbridge, T. C. (2000). What knowledge is important to a software professional? *Computer, 33*(5), 44-50. https://doi.org/10.1109/2.841783

Miller, C. S., & Dettori, L. (2008). Employers' perspectives on it learning outcomes. In *Proceedings of the 9th ACM SIGITE Conference on Information Technology Education, ser. SIGITE '08*. New York, NY, USA: ACM, 213–218. https://doi.org/10.1145/1414558.1414612

Mishra, A., Ercil Cagiltay, N., & Kilic, O. (2007). Software engineering education: Some important dimensions. *European Journal of Engineering Education, 32*(3), 349-361. https://doi.org/10.1080/03043790701278607

O'Flaherty, J., & Phillips, C. (2015). The use of flipped classrooms in higher education: A scoping review. *The Internet and Higher Education, 25*, 85-95. https://doi.org/10.1016/j.iheduc.2015.02.002

Paez, N. M. (2017). A flipped classroom experience teaching software engineering. In *Proceedings of the 1st International Workshop on Software Engineering Curricula for Millennials*, IEEE Press, 16-20. https://doi.org/10.1109/SECM.2017.6

Pieper, J., Lueth, O., Goedicke, M., & Forbrig, P. (2017, April). A case study of software engineering methods education supported by digital game-based learning: Applying the SEMAT essence kernel in games and course projects. In *Global Engineering Education Conference (EDUCON),* 2017 IEEE, 1689-1699. https://doi.org/10.1109/EDUCON.2017.7943076

Rana, M. E., Saleh, O. S., & Ghazali, O. (2017). Cloud based software engineering learning environment: Guidelines to host software engineering tools on the cloud. *Journal of Theoretical and Applied Information Technology, 95*(3), 525.

Radermacher, A., Walia, G., & Knudson, D. (2014). Investigating the skill gap between graduating students and industry expectations. In *Companion Proceedings of the 36th International Conference on Software Engineering,* ACM, (291-300). https://doi.org/10.1145/2591062.2591159

Reddy, Y. R., & Nori, K. V. (2014, April). Teaching software product engineering in undergraduate computing curriculum. In *Software Engineering Education and Training (CSEE&T)*, 2014 IEEE 27th Conference, 175-178. https://doi.org/10.1109/CSEET.2014.6816798

Reza, S., & Baig, M. I. (2015). A study of inverted classroom pedagogy in computer science teaching. *International Journal of Research Studies in Educational Technology, 4*(2), 19-30. https://doi.org/10.5861/ijrset.2015.1091

Ross, D. T., & Schoman, K. E. (1977). Structured analysis for requirements definition. *IEEE Transactions on Software Engineering, SE-3*(1), 6-15. https://doi.org/10.1109/TSE.1977.229899

Shafto, P., Goodman, N. D., & Griffiths, T. L. (2014). A rational account of pedagogical reasoning: Teaching by, and learning from, examples. *Cognitive Psychology, 71*, 55-89. https://doi.org/10.1016/j.cogpsych.2013.12.004

Shaw, M. (2005). Software engineering for the 21st century: A basis for rethinking the curriculum. *Technical Report CMU-ISRI-05–108*.

Tai, C. H., Leou, S., & Hung, J. F. (2015). The effectiveness of teaching indigenous students mathematics using example-based cognitive methods. *Journal of Interdisciplinary Mathematics, 18(*4), 433-448. https://doi.org/10.1080/09720502.2015.1023547

Tillmann, N., De Halleux, J., & Xie, T. (2011, May). Pex4Fun: Teaching and learning computer science via social gaming. In *Software Engineering Education and Training (CSEE&T),* 2011 24th IEEE-CS Conference, 546-548.

Varma, V., & Garg, K. (2005). Case studies: The potential teaching instruments for software engineering education. In *Quality Software, 2005 (QSIC 2005). Fifth International Conference*, IEEE, 279-284.

Yadav, S. S., & Xiahou, J. (2010, June). Integrated project based learning in software engineering education. In *Educational and Network Technology (ICENT)*, 2010 International Conference, IEEE, 34-36.

APPENDIX

SCHOOL OF ENGINEERING, CUSAT
STUDENT OPINION SURVEY

Declaration by the Researchers

This opinion survey is conducted to collect the students' feedback about new teaching strategy carried out for teaching software engineering. The data collected will be used only for research purpose.

Name of degree: M.Tech- Software Systems Semester and Year: S1, 2017

(Put tick mark in appropriate position .If your answer is '**Yes'**, write the benefits/outputs you attained. If your answer is '**No'**, write down the hindering factors.)

1. Have you studied Software engineering/equivalent courses at undergraduate level? Yes /No
   If 'yes', point out the factors that you have experienced with the new teaching strategy in the advanced level?
2. Have you completed your out-class activities on time? Yes / No
3. Whether the learning materials shared for out-of-class learning was appropriate and helpful in completing the in-class activities? Yes /No
4. Do you think the project artifacts (SRS, SDD, and Test Plan) created helped you to enhance your practical knowledge? Yes/No
5. Whether the activities behind the creation of project artifacts improved your skills to work in a project context within a team? Yes/No
6. Do you think the individual presentations helped you to improve your presentation and communication skills? Yes/No
7. Whether the examples used by the instructor in the class room were appropriate and helpful to understand the concepts clearly? Yes/No
8. Do you feel the new teaching strategy (Teaching by-flipped classroom, examples, project artifacts, and presentations) is beneficial when compared with traditional teaching? Yes/No

BIOGRAPHIES

Jeevamol Joy is a Research Scholar at Cochin University of Science and Technology, Kerala, India. She earned her undergraduate degree from Mahatma Gandhi University, India and master's degree from Manonmaniam Sundaranar University, Tamil Nadu, India. She worked as an IT System Analyst in Retail domain for 8 years at UST Global, Technopark, India. Her research areas include project-based learning, recommender systems and e-learning.

Renumol V G, PhD, is a Professor and Head of the Department of Information Technology at Cochin University of Science and Technology, India. She has earned her undergraduate and postgraduate degrees from the same University. She received her Ph.D. from Indian Institute of Technology, Madras, India and received Post-Doctoral Fellowship from Indian Institute of Technology, Mumbai, India. Her research area includes Cognitive Psychology in Education, Computing Education, Educational Technology and ICT in Special Education.