# LEARNING THROUGH PRACTICE VIA ROLE-PLAYING: LESSONS LEARNT

**Sabrina Ahmad, Emaliana Kasmuri, Noor Azilah Muda, Azah Kamilah Muda**

*Universiti Teknikal Malaysia Melaka (MALAYSIA)*
*sabrinaahmad@utem.edu.my, emaliana@utem.edu.my, azilah@utem.edu.my, azah@utem.edu.my*

## Abstract

Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machine. Sound software engineering closely related with socio-technical activity that depends on several human issues which are communication, collaboration, motivation, work environment, team harmony, engagement, training and education. These issues affect everything for students to fully understand software engineering and be prepared for software development careers. Therefore courses offered in the university must also consider the sociological and communication aspects, often called the socio-technical aspects. One popular method is to use role-playing exercises. Role-playing is a less technologically elaborate form of simulation for learning interpersonal skills and is analogous to rehearsal. It is particularly helpful when students are having difficulties to relate lessons learnt in the university to the applicability of the knowledge in the real implementation. This is because many students view software engineering as meaningless bureaucracy and have little interest in the knowledge delivered in the lecture hall. This scenario impedes the expansion of current knowledge and inhibits the possibility of knowledge exploration to solve range of industry problems. Simply lecturing about software engineering will never engage students or convince them that software engineering has value. Given this student bias, the goal of teaching software engineering often becomes convincing students that it has value. To achieve this, students need to experience firsthand the sociological and communication difficulties associated with developing software systems. In this paper, we argue that in teaching software engineering we must cover two essential things; delivery of knowledge and skills required in the software engineering domain in a form of lecture and hands-on practice to experience the value of the knowledge and skills learnt. We report on our experiences gained in deploying role-playing in master degree program. Role-playing is used as pedagogical tool to give students a greater appreciation of the range of issues and problems associated with software engineering in real settings. We believe that the lessons learnt from this exercise will be valuable for those interested in advancing software engineering education and training.

Keywords: Role-playing, higher education, software engineering.

## 1 INTRODUCTION

University education plays a dominant role in shaping principles and values of the field. In a field as rapidly growing as software engineering, the principles and values are being shaped jointly by university and industry influences. This is due to the commonalities and differences of underlying theory and industry practice in software engineering. Since information processing has become an essential part of the way society manages their life and also a key to industrial power, the need of having an appropriate software engineering education is crucial.

### 1.1 Software Engineering Education

It is a challenging endeavour to design software engineering curriculum, especially to computer science majors. Computer science curriculum tends to focus on the technical side of the field whereby students spend most of their time in courses about theoretical and programming aspects of computer science. This often led to a stereotype mind set of thinking programming is the most important component of the degree. By the time students arrive at an upper division of software engineering course, they often believe that writing code is all it takes to build a large software system. Thus, students have a difficult time believing that software engineering topics are important. The high level software engineering courses such as project management, software architecture and requirements

engineering are always seen as theoretical knowledge with minor impact to the real practice in industry [2]. This is because simply lecturing about software engineering will not engage students or convince them that software engineering has value. Given this student bias, the goal of teaching software engineering often becomes convincing students that it has value. To achieve this, students need to experience firsthand the sociological and communication difficulties associated with developing software systems.

## 1.2 Active Learning

Software engineering cannot be taught exclusively in the classroom. This is because software engineering is a competence and not just a body of knowledge. Any presentation of principles or experience that is not backed up by active and regular participation by students in real projects is sure to miss the essence of what the students need to learn.

Donald Schon [5] presents evidence that experts in a range of professions from architecture to psychoanalysis exhibit what he calls reflection-in-action. Expertise, according to Schon, is the interplay of two competencies; core competencies that permit the practitioner to act respond effectively in familiar problem situations, and reflective skills that let the practitioner reasons about his or her skills and knowledge when the most immediate course of action seems likely to be unsuccessful. Translated into software engineering, Schon distinction is between the type of competence that a designer uses when making design decisions and the type of competence that leads to reason about the design method itself. Skills of the first type can be taught through lecture and by applying through small exercises. On the other hand, skills of the second type are extremely difficult to teach by instruction, because their effective deployment depends on the practitioner being sensitive to a wide range of contextual effects; some of them are not within the field of engineering at all.

Educating this awareness, and knowing when to use a rigorous technique and when to trust one's instinct, is something that can only be learned through experience. Typically, a student's first experience on software development project is via an intern position or his/her first full-time position. However, prior exposure to the corporate project environment would greatly improve a student's performance in industry. In order to develop students for successful careers in software engineering, specifically for software development, they must be immersed not only in the software development lifecycle and paradigms, but also in the workings of project teams.

Following Introduction, Section 2 explains the existence of variety of pedagogical tools for higher education and how role playing fit in into the curriculum. This is followed by Section 3 which discusses curriculum model with role playing elements. Section 4 elaborates lessons' learnt from the perspective of students and Section 5 is the conclusion.

# 2 ROLE-PLAYING AS A PEDAGOGICAL TOOL

The pedagogy of education often refers to the strategies of education or instructive strategies [4]. In outcome based education (OBE), the instructors have the freedom to choose and select the contents and methods of teaching to help the students achieve the learning outcomes of the subjects. There are several methods of delivering knowledge to students and among them are Problem-based Learning (PBL) and Cooperative Learning (CL).

Problem-based learning is a pedagogical tool that concentrates on ill-structured problems. The problem to be solved usually is a complex question, task, or issue that needs resolution through inquiry. It is a realistic problem in a relevant context, especially if the knowledge gained is to be applied later. Solution of the problem usually includes a range of alternatives that are established through the application of new knowledge and reasoning, not just simple formulas. The role of the instructor is to guide and facilitate, rather than as the source of knowledge or as an information-giver [3].

Cooperative learning on the other hand is a method of instructing students to work together in groups, usually with the goal of completing a specific task. This method can help students develop leadership skills and the ability to work with others as a team. The five basic elements of cooperative learning as described by [6] are:

a) Positive Interdependence

b) Individual and Group Accountability

c) Interpersonal and Small Group Skills

d) Face-to-Face Promotive Interaction

e) Group Processing

Compared to problem-based learning, cooperative learning method is seen to be more appropriate to suit the content of Software Engineering courses in classrooms. Not only because the course itself is too broad and abstract, the technique used to attract students to participate and delivers the knowledge should also be creative and interesting. One of the techniques often used in the cooperative learning is the collaborative inquiry or also known as role playing, which highlights the distribution of roles leads to interdependent as well as individual responsibility with a team. Role play is a method where someone rehearses situations for future performances and to improve his or her ability within a role. It is also a less technically elaborate form of simulation to learn interpersonal skills and need not much of rehearsal in order to be applied [7].

As described by [2], Software Engineering field is a difficult and has more theoretical elements than the technical elements. Because of its nature, it is hard to deliver the related subjects in classrooms and often, the goal of delivering the knowledge becomes convincing student that the field has its own value. Observing this, the adoption of role-playing method is appropriate where the method is applied to let the students experience themselves in learning and valuing the contents of the subjects related to the broad topic of Software Engineering. By letting the students take roles as the key persons involved in simulating the software engineering project, they will experience all the sociological and communication difficulties associating with the real world software system development. While facing the difficulties, the students involved will creatively impersonate the roles they are playing to find resolutions to the facing problems. This indirectly will give them firsthand experience to face the reality of real world environment in software engineering field.

To further explain the role playing method, Software Engineering subject is chosen to simulate the subject contents where the students act as developers and end-users. The simulation process is to capture requirements from end users where the outcome of this simulation is the requirements specification report or Software Requirement Specification (SRS). In the role-playing method, students are divided into two groups: a) developers (programmer, system analyst and data administrators) and users (stakeholders and end-users). The developer captures the requirements from users by using several methods that are interviews, observations and document analysis. The users on the other hand act as individuals or interviewees who will explain the current business processers and flow of the current system. This is to make the developers understand the current business flow and then can capture the requirements of the to-be developed system.

In the role-playing process, the students involved are given the guidelines and instructions from the instructor and the outcomes of the method are clearly explained. With the guidance from the instructor, the students will take part in the simulation with the objective of producing the SRS document as the outcome of the role-playing method.

## 3 THE CURRICULUM MODEL

The curriculum structure in Centre for Advanced Software Engineering (CASE) is divided into three major structures, which are delivery of modules, industrial trainings and final project (Figure 1). The duration to complete the curriculum is 18 months. The structure covers both theoretical and practical parts of software development process.

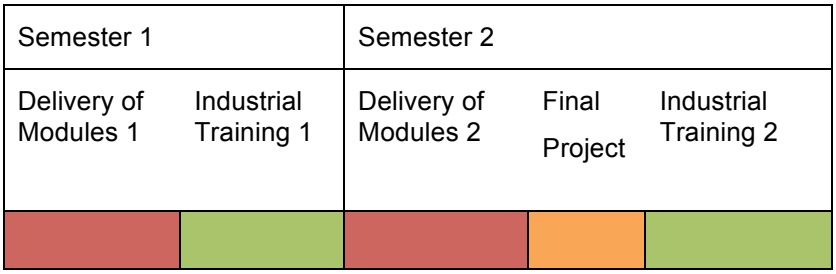


Figure 1: Curriculum Structure for MSc (Real Time Software Engineering) at CASE

## 3.1 Delivery of Modules

The structure of modules delivery covers the whole life cycle of software development phases including its supporting activities. Each modules represent individual phases in the software development and the supporting activities which includes software requirement and analysis, software design, software implementation, software configuration, software testing, project management and software quality assurance. The duration for each module is approximately one week. The duration was sufficient to cover end-to-end content for each module. The modules cover theoretical practical (hands-on exercises) aspects for each phase in the software development life cycle. In the end the students have to sit for their final exam and pass the papers to prove their sound theoretical knowledge in software engineering.

## 3.2 Final Project

The students need to develop a final project entitled On Board Auto-Cruise System. The project is to develop a software module to simulate an auto cruise system of a car using object oriented as the selected software development approach in AIX environment using C++. The project is using RUP, UML and DoD 2167-A. The students are divided into groups with 5 team members for each. Each team members is given a role that they have to play in order to complete their project. The role includes one project manager, one system analyst, one software configuration engineer, one software quality engineer and two software developers Figure 2. The group is given three weeks to complete the project. Each team is having a designated space as their office complete with workstations, tools and stationeries. The team has to adhere to the official working environment standard including attire, clocking their attendance and applying for leave if necessary.

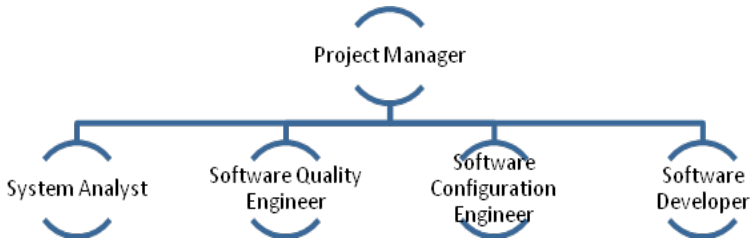


Figure 2: Team Structure for Final Project

Project Manager plans and oversees the whole development, assign task to the members, keep track the progress of the project, chaired meetings with the client and review session and manage the team spirits and synergy. Short meetings are regularly held to update the progress of the development by each team member and to solve any issues rose during the development. The Project Manager documents the project plan into Software Development Plan (SDP).

System Analyst gathered and analyzed the requirements for the system. The person carries this duty is responsible to model and design the structure of the system with the assistance from the programmers and software quality engineer. The findings and results from analysis and design are documented in Software Requirement Specification (SRS) and Software Design Document (SDD).

Software Developers are responsible to implement the module according to the design in SDD. The programmers are responsible to draw the algorithm for the each module, program the algorithm, unit-test the program, debug and integrate the program. Any major changes to the design will be inform to the System Analyst before updating the SDD. The programmers must ensure their source codes are well documented, reviewed and item configured.

Software Configuration Engineer controls all software artefacts produced by the team that includes SDP, SRS, SDD and source codes using configuration management tool. Artefacts are version controlled and base lined. Findings from discussions, meetings and reviews are properly documented and labelled into separate folders in softcopy and hardcopy. Software artefacts, which are labelled as confidential, are accessible to the team members only and are not for public circulation. Software Change Request is drawn by this person to manage any request of change for the client.

Software Quality Engineers to document the test plan and test result using Software Test Plan (STP) and Software Test Result (STR). The person who carries out this role is also responsible to ensure all software artefacts comply with the standard used in the project.

Based on the activities carried out for each phases and supporting activities as whole in the project, the team is practicing CMM Level 3.

The team is a dynamic team where the role is interchangeable when it is required. For example, during implementation all team members put on programmer hat and get their hands into C++ to complete the program. Through this project and dynamic team structure, the students experience all phases and supporting activities in software development life cycle. The students found this project has strengthens their knowledge and enrich their experiences.

## 3.3 Industrial Training

At the end of each semester, the student will have to complete their industrial training. The students are assigned to companies that practice software development process. During the industrial training period, the student will apply and practice the knowledge gained in the class to complete the software development activities assigned to them by their industrial supervisor. Most of the companies appreciate the technique or method introduced by the students and continues adopting it into the company development methodologies.

# 4 LESSONS' LEARNT

This section reports on some of the significant lessons learnt while undergoing this curriculum. The aim is to share the value gained as students with academics and practitioners that are interested in designing and delivering SE courses at the tertiary level. Two of the authors of this papers were students at CASE who undergone the process.

While other papers are reporting the lessons learnt from the perspective of knowledge provider or the Professors [1, 2, 5], this paper is reporting the lessons learnt from the perspective of students who experienced the teaching and learning process. Based on the curriculum model explained in Section 3, the role play is implemented in the Final Project for each semester.

## 4.1 Environment

During the Final Project duration, several classrooms and computer labs were set up to imitate the environment of software development working space. The students' groups were assigned their dedicated working space to be utilized in order to deliver the tasks given. The classroom was divided into several clusters and the computer labs were partitioned to accommodate all the groups. Each group gets enough basic furniture like tables, chairs and a white board for working space and shared lounge for informal meeting space. Also, all groups get computers for the purpose of documentation, development and configuration management. In addition, all groups were responsible to manage their own resources such as papers, stationeries and document folders.

From the perspective of students, the environment set up was successfully creating the atmosphere of software development anxiety and belonging to the team. Even though it was not a contest, the competitiveness among the groups exist to meet the milestones according to the timeline given and to deliver high quality deliverables. On top of that, the dedicated working spaces given were fully utilized to protect the intellectual property of project deliverable items and progress thus far.

## 4.2 Experience and Values

As explained in Section 3, the role of being part of the software development team was assigned to the students by the lecturers. Therefore, the students need to understand the task of being a project manager or a system analyst or a quality manager or a developer. Inexperience, students tend to try hard in order to fulfill the responsibility of the role given. For example, each team was responsible to protect the confidentiality of some of the documentations. Therefore, the students need to archive, label and keep all related files accordingly to ensure sufficient action has been taken to protect the project's property.

The process and the progress of software development were not always ideal. At this stage, the students realized that theory discussed in lectures and books were not always the case. Even though

the theory provides guidance, individual students still need to take action according to the current situation. If a student played a role of a project manager, as an example, the student needs to make decision in which will give direct impact to the project in hand and the team as a whole. The most important value obtained from this experience was dealing with variety of people with different background, competency and the way they think. The element of communication and interpersonal skills are as important as technical skills to ensure success. In addition, working together as a team is not easy when it comes to applying software engineering knowledge in order to develop a workable system. Even harder, the development process must adhere to DoD 2167-A standard.

In accordance to fulfilling DoD 2167-A standard, the versioning was taken care of seriously. Configuration management handled the development progress and software documentation updates systematically. Overwhelmed with piles of many versions of software documents produced by system analysts, model designs produced by designers, letters and request change forms from the customers and many other related artefacts, forward and reverse traceability were surprisingly at ease. This was the moment when students appreciate the bureaucracy of software engineering. The same spirit was brought to the industry during the industrial training period at the end of each semester. Most of the students put effort to let the industry sees the benefits of official procedure of software engineering. Software engineering process was meant to ease the software development process and not the other way round.

## 4.3 Motivation and Appreciation

The motivation and appreciation of the values gained during the study mould the shape of brand new practitioners and academics in the field. Graduates from CASE who experienced the process through role-playing understand the underlying motivation of software engineering good practice. When knowledge is told or feed to the students, it will be forgotten but when it is experienced and appreciated, it stays. As practitioners, the possibility of applying software engineering good practice is huge as they already experienced the benefit of undertaking it. As academics, the experience will probably influence the improvement of curriculum design and the method of teaching and learning in other universities.

# 5 CONCLUSION

Role-playing is found to be an effective approach for the students to learn and grasp the knowledge in software engineering. Through role-playing the students has found software engineering subject very interesting. They have the chance to put practical knowledge into practice through the role that was assigned to them in the project. This experience has enriched them and provides clearer view about the happenings in software development. Thus, they appreciate the knowledge and experiences and continually practicing and apply this when they join the software development industry. This approach is proven through the feedback given by their industrial training supervisors and company supervisors.

# REFERENCES


[1] Al-Ani, B. and Yusop N.(2004). Role-playing, Group Work and other Ambitious Teaching Methods in a Large Requirements Engineering Course. Proceedings of the11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, pp. 299-306.

[2] Henry, T. R. and LaFrance J. (2006). Integrating Role-play into Software Engineering Courses, Journal of Computer and Small Colleges, Vol 22 (2) pp. 32-38.

[3] Levin, B.B., Hibbard, K., & Rock, T.T. (2002). Using Problem-based Learning as a Tool for Learning to Teach Students with Special Needs. Teacher Education and Special Education, 25 (3) 278-290.

[4] Reigeluth, C.M. (1999). What is Instructional Design Theory? In *C.M. Reigeluth (Ed.) Instructional design theories and models: A new paradigm of instructional theory* (Vol. 2, pp. 5-29). Manwah, NJ: Lawrence Erlbaum Associates.

[5] Schon, D. A.(1987), Educating the Reflective Practitioner: Toward a New Design for Teaching and Learning in the Professions, Jossey-Bass Inc., San Francisco, California.

[6] Yusof, K. M., Hassan S. A. H., Jamaludin M., Z., Harun N.F. (2011). Cooperative Problem-based Learning (CPBL) . In Proceedings of IEEE Global Engineering Education Conference, Amman, Jordan.

[7] Zowghi D. and Paryani S.(2003). Teaching Requirements Engineering through Role Playing: Lessons Learnt. Proceedings of the 11th IEEE International Requirements Engineering Conference, pp. 233-241.