THE EFFECTIVENESS OF SIMULATION EDUCATION FOR UNDERGRADUATE STUDENTS IN SOFTWARE ENGINEERING AREA

Sabrina Ahmad, Noor Azilah Muda, Azah Kamilah Muda, Intan Ermahani A. Jalil

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

Abstract

Simulation in education is a replacement model for real world experience. This approach is seen effective to give undergraduate students the opportunity to apply their theoretical knowledge and practice their skills learnt in laboratory. Simulation education is designed to emphasize outcome based education or known as OBE to bridge the gap between theory and practice. Implementing simulation model to represent real world practice in industry, undergraduate students are trained to deal with real problems in the right atmosphere. The effort eventually prepares the students to face a real working environment when they are graduated from the university. This paper discusses the curriculum designed for simulation education applied for undergraduate students in university, its implementation and the analysis of the outcome product. A specific case of Universiti Teknikal Malaysia Melaka is presented here.

Keywords: Simulation education, outcome based education.

1 INTRODUCTION

Simulation used in education has potential in generally two areas. The common exercise is a replacement model for real world experience. It has many advantages especially not putting real world elements at risk and save significant amount of cost [3]. The other area is exploring simulation model in which a student can learn underlying theories based on exploration. This paper presents the implementation of the first area which is replacing a real world experience through simulation in software engineering education for undergraduate students in university.

1.1 Replacing Real World Models

Simulation education is popular in medical school as they can replace human subjects and take them out of hands of the inexperienced. By doing this, medical students have the opportunity to practice a certain procedure or general diagnosis dozens or even hundreds of times. The vast number of runs and data that can be gathered by simulations is a quality that has been hailed about them. Replacing humans for safety reasons is not the only advantage as research done revealed a fascinating statistic claiming students who learned by simulations completed their real operations 29 percent faster and hesitated less during the process.

When future aircraft pilots learn their profession, there is a similar situation; they need to know how to behave in dangerous situations, but nobody would like to expose them to such risk just for the sake of training. Therefore, airlines provide powerful simulators that model the whole aircraft and its environment for the pilot. By using these simulators, all difficulties can be experienced without any risk.

When future doctors and pilots can benefit from simulation education, the same approach can be applied for educating software engineers. This is because the key successful education is motivation. For example, if the information of free source code is mentioned in lecture, most students will listen carefully and apply the knowledge immediately because they are affected by it. However, if the theory of software engineering is delivered, the students will hardly listen, because they cannot imagine that there is a real problem. They have developed lots of rather small programs on their own, and they expect the same kind of difficulties when a team does the same kind of work. Therefore, the message of software engineering process vitality to software development will not really be appreciated. The best motivation for learning software engineering is experiencing by undergoing software project development process that requires multi-skills on top of technical skills such as project management and communication skills [1]. After students have experienced the process themselves and most probably facing obstacles, they will start asking questions, and they will listen to the answers. Then the ground is ploughed and ready for seeding.

1.2 Learning by Doing in Software Engineering Education

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. This is supported by Denning [2] who said that computer science and engineering degrees should be based at least in part on demonstrations of accomplishments and competencies.

Donald Schon [4] 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.

2 THE CURRICULUM DESIGN

Faculty of Information and Communication Technology, University Teknikal Malaysia Melaka is offering a subject called Workshop II which implementing simulation education. This subject is a compulsory subject for all third year undergraduate students. The main purpose of this subject is to expose the students to the team working environment, team project management such as risk management and time management, leadership and critical thinking to complete the project in a timely manner according the system development lifecycle. The students also need to do research on finding or using new technology in the implementation of their work and to use existing knowledge to carry out their work.

The subject is introduced to the third year students because Workshop II simulates the contents of several subjects learned in previous year of studies. Among the subjects are the programming, project management, software engineering, requirements analysis and design, database, and database design. The students need to apply all the knowledge and experiences learned from previous subjects in order to complete the task and present the end product to the faculty. Fig 1 illustrates the mapping overview of the subjects that each student should undertaking prior to the development of the Workshop II project.

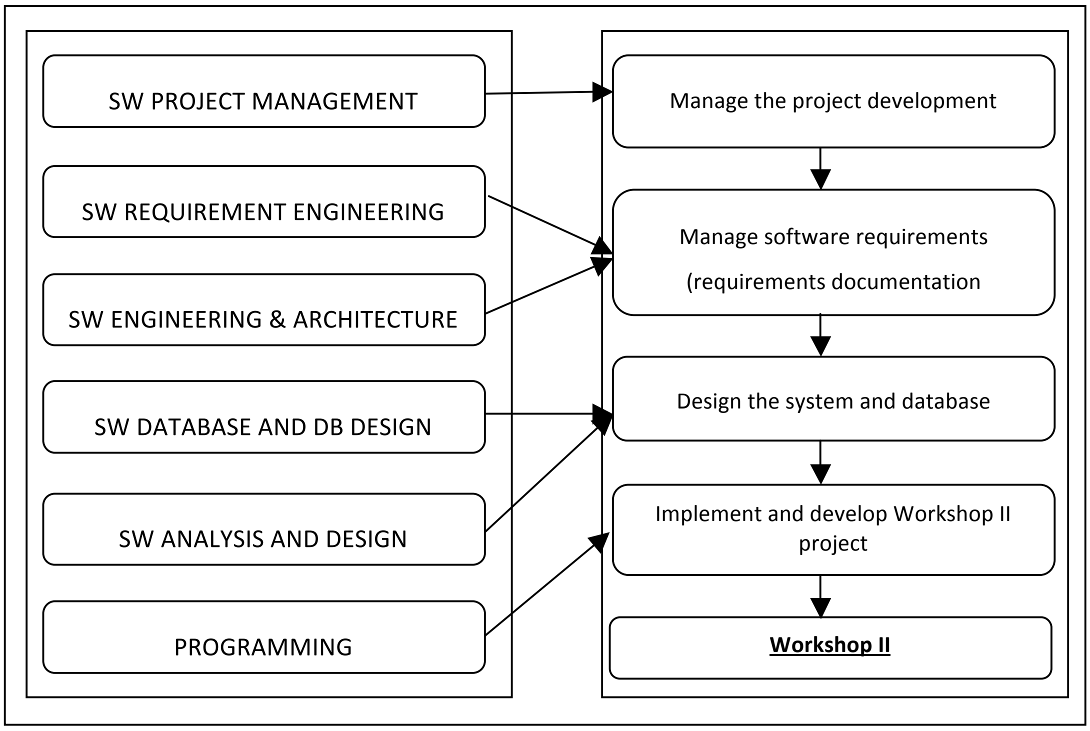
SW PROJECT MANAGEMENT
SW REQUIREMENT ENGINEERING
SW ENGINEERING & ARCHITECTURE
SW DATABASE AND DB DESIGN
SW ANALYSIS AND DESIGN
PROGRAMMING
Manage the project development
Manage software requirements (requirements documentation
Design the system and database
Implement and develop Workshop II project
Workshop II

Fig 1: Subjects mapping to the stages of Workshop II development

Based on Fig 1, the left side squares show list of subjects that directly mapping to the stages of project development in Workshop II. The arrows show which subject or subjects that contribute the knowledge needed to perform each stage of the project implementation. It is very important for the students to learn the subject contents following the sequences of curriculum offered by the faculty to make sure they are prepared accordingly to complete Workshop II. These are knowledge that can be delivered by taught but reflective thinking as mentioned in Section 1 must be develop through experience by enforcing practice. This is the reason why simulation education is vital to the software engineering curriculum.

A part of software development environment in which the students are applying technical and soft skills to develop software project, other deliverables like requirements documentation and models are a part of the curriculum outcomes too. The students must complete and submit a report to the supervisor. In the design phase, they must ensure that it covers conceptual design, logical design and physical design. The conceptual design is an abstract design that contains the essential components and entities. The main purpose of the conceptual design is to provide an overview of the proposed settlement. Components can represent technology systems or external systems that require integration or data stream as a whole, or perhaps only a function of the system. Logical design is more detailed than the conceptual design because it is composed of components and relationships. The design consists of the business logic, application name and description. A physical design on the other hand consists of the components and entities that are identified during the application requirements gathering, the location environment and the service-specific software.

In preparing the final documentation of Workshop II, students should follow the guidelines of report writing where the contents contain the architecture of the application, the database design (Entity Relationship Diagram or ERD, data normalization and data dictionary), interface design and related modules and sub-modules (if any). The explanations of the relationship between interface designs for each module need to be included as well.

The report should also explain all the implementation of the functions for each developed module. At this stage, any relevant diagrams illustrating the architecture application environment (for example, deployment diagram, client-server configuration, IP addresses and network) are important for better evaluation from the supervisor point of view. Other important aspects of the report are the software configuration management system, the environment configuration, system development environment, version control procedure, feature-safety elements (including the use of firewalls, encryption, SSL for web based application, and the correct data ranking performance through the correct IP)

At the end of the report, the students should conclude the project by summarizing the results of the implementation phase related to the implemented application. For the final delivery, each group of the students should present and demonstrate the functional modules where evaluation focuses on the algorithms used, memory location, function reusability, appropriate file names, the use of indentation and comments, proper error handling, security features, and testing (Unit testing with "test cases" and Module testing with "test cases"). The assessment also includes the creativity, skills, efficiency and appropriateness of the project. The improvement of the system based on the customer's comments is evaluated too.

The expectations of this Workshop II are:

a) Students can analyze and develop a software project in a team.
b) Students can apply the concept of development and system design in the project implementation.
c) Students can identify, analyze and manage changes to project scope throughout the project lifecycle.
d) Students can manage projects in an ethical group.
e) Students can present and defend the work of their project.

3 THE IMPLEMENTATION

As Workshop II is designed to develop understanding of the collaboration between multiple specialized fields in general, this simulation approach teaches the cross-discipline collaboration. A part of essential technical skills, - to elicit and to analyse the requirements, to design the architecture and the software system, to deploy the design, to develop and to test the system - other skills such as critical thinking, project management, risk management, communication, negotiation and leadership is vital to the software project success too. This section explains the execution of the workshop which revealed the skills developed throughout the process.

3.1 The Workflow

Workshop II is treated as a subject and run throughout a semester. Instead of a lecturer is responsible to teach a subject, a committee of six lecturers are appointed to monitor and to manage the implementation of the workshop. This committee is responsible to group the students into four each group and assign them a supervisor. The member of each student's group is selected based on their academic achievement and a mixture of high achiever to the less competent students is ensured. Every group member plays a role as a systems stakeholder such as project manager, system analyst, software engineer, designer, software architect and developer. Each group is responsible to identify the role of member in their group and clarify the tasks need to be fulfilled in the proposal. The element of individual log book is imposed to ensure that the tasks described in the proposal are delivered.

Explained in Fig 2 is the workflow on the implementation of Workshop II. Rectangle notation in the diagram describes working tasks need to be done by the students. Besides, parallelogram notation state deliverables need to be submitted throughout the semester.

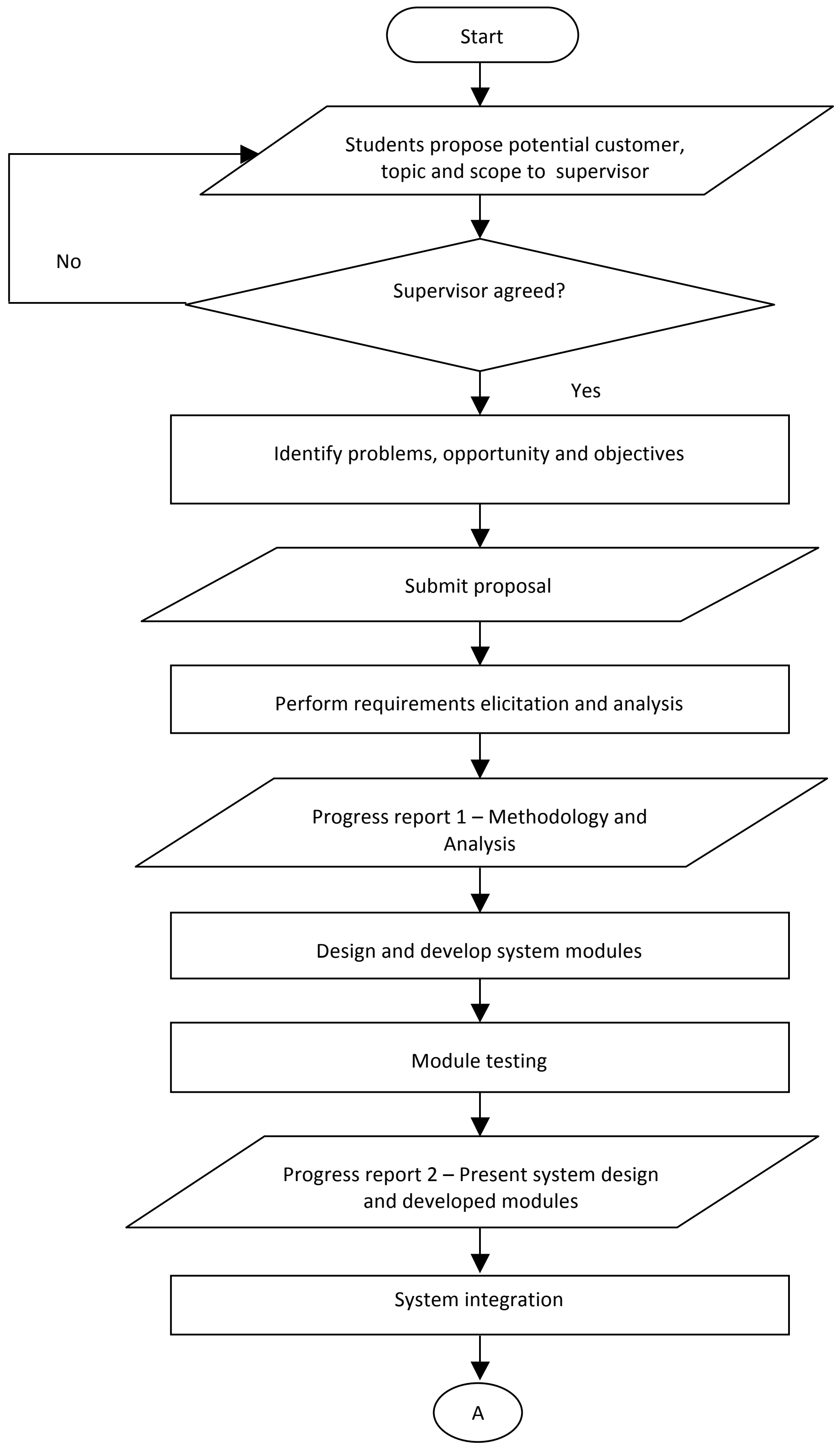
Start
Students propose potential customer, topic and scope to supervisor
Supervisor agreed?
No
Yes
Identify problems, opportunity and objectives
Submit proposal
Perform requirements elicitation and analysis
Progress report 1 – Methodology and Analysis
Design and develop system modules
Module testing
Progress report 2 – Present system design and developed modules
System integration
A

Fig 2. (a) Flow chart for Workshop II Implementation Part 1

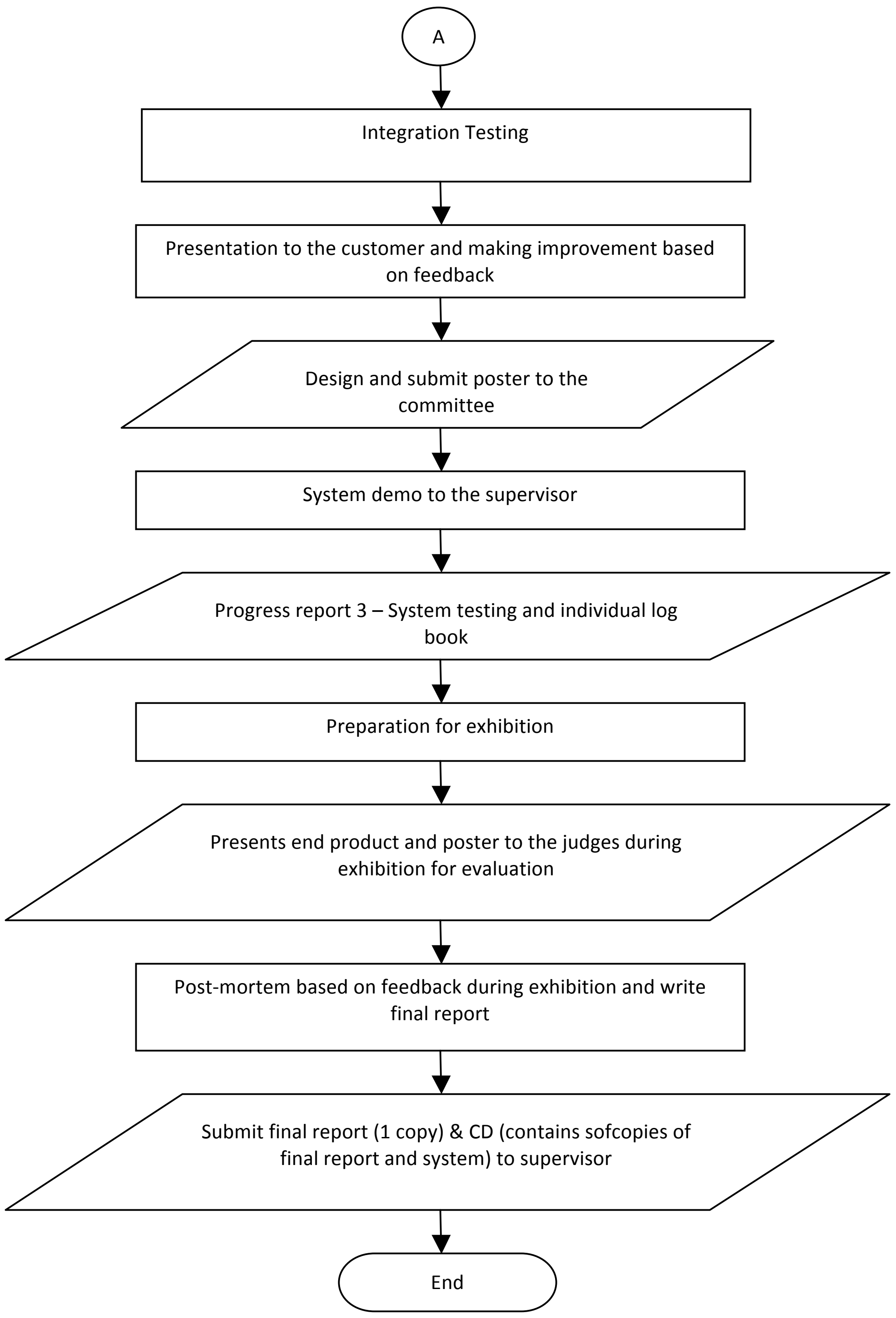
A
Integration Testing
Presentation to the customer and making improvement based on feedback
Design and submit poster to the committee
System demo to the supervisor
Progress report 3 – System testing and individual log book
Preparation for exhibition
Presents end product and poster to the judges during exhibition for evaluation
Post-mortem based on feedback during exhibition and write final report
Submit final report (1 copy) & CD (contains sofcopies of final report and system) to supervisor
End

Fig 2. (b) Flow chart for Workshop II Implementation Part 2

The unique element in the Workshop II implementation is the real customer who has real problem and demand real solutions. The involvement of the customer is not only during the earlier stage of eliciting requirements but also towards the end of the process. The customer needs to confirm if their requirements are correctly translated into a working system and if necessary provides comments to further improve the system. Another unique element in this workshop is the opportunity for the students to display and to demonstrate their end product to the university community. Therefore, an exhibition format is imposed here. Each group needs to prepare a poster that explains the system and the development process. They also need to demonstrate the system to the judges appointed by the faculty during the exhibition. At the end of the process, a proper report based on simplified version of standard documentations which are software development plan (SDP), software specification document (SRS), software design document (SDD) and software test document (STD) is produced. The detail of the report content is explained in Section 2.

4 THE EVALUATION AND DISCUSSION

This section discusses students' achievement after undergoing Workshop II. One semester results is presented here to show the value of knowledge and skills designed to be applied and experienced from it. This is shown in the items evaluated throughout the implementation of Workshop II.

In the previous semester, 114 students were divided into twenty-eight software development team. However, this doesn't mean that the grade is team based since the evaluation is divided into two section; individual (30%) and group (70%). Unlike common subjects which are evaluated by final exams, assignments, quizzes and tutorials, Workshop II combined the most important element in software engineering curriculum and thoroughly evaluated based on six parts; which are proposal (5%), methodology and system analysis (10%), design and implementation (15%), testing and log book (20%), final report (20%) and demonstration during exhibition (30%).

The first part is the proposal. It allows students to show their creativity in managing project team with specific task to determine deadlines, establish a working project title, describes the business process of the proposed project, observing existing system in order to identify the uniqueness of the proposed system and a brief software and hardware requirements.

The second part is the progress report 1. Here, students are evaluated based on the framework to be used in structuring, planning and control the process of developing a system or software. The justification of the chosen software development methodology must be provided and explanation of each process must be presented in a report. The task of identifying a software development methodology is crucial because the best approach in applying a methodology is considered as managing software development risk. Students are also evaluated based on System Requirement Specification (SRS) document as an output for system analysis task. Besides, student should demonstrate ability to deploy system investigation, identifying problems or using the information to recommend improvements to the current system. On top of that, the students are assessed on the ability to distribute task for group members and other project management elements will be evaluated as well.

The third part is progress report 2. System Design Documentation (SDD) is evaluated here. After the purpose and specifications of software are determined in system analysis, the students will design a plan for problem solution. It involves the analysis, design, and configuration of the necessary hardware and software components such as ERD diagram and GUI, to support the solution's architecture of the proposed software. Later, students are evaluated through their implementation stage in which software engineers actually program the code for the project.

The fourth part is progress report 3. Here, students are evaluated on the ability to test the system based on test cases designed. The interface and the computer-human interaction are evaluated too. Finally is the final report in which detail evaluation is done on every section as mentioned in Section 2.

Fig 3 shows the grade obtained by students who undergoing Workshop II in one semester. The learning process through simulation education like Workshop II ensures that the students acquire a certain level of experience and skills which can only gain through practice. This conveys the idea that students with at least credited grade are able to perform such skills and trained themselves in practising to reason out when dealing with the real software development problems. The results pattern for each semester is more or less the same as most of the students are able to score good grades. However, it is important to emphasize here that it takes huge effort and high commitment to score such grades. The difficulty is reflected in the detail items in the evaluation criteria. The unique elements of real customer and exhibition format to demonstrate the final product motivates the students to prove that they are knowledgeable, skilful, reliable and accountable to undertake such responsibility. This shows that simulation education applied in Workshop II has served as a good platform for students to experience the software development process.

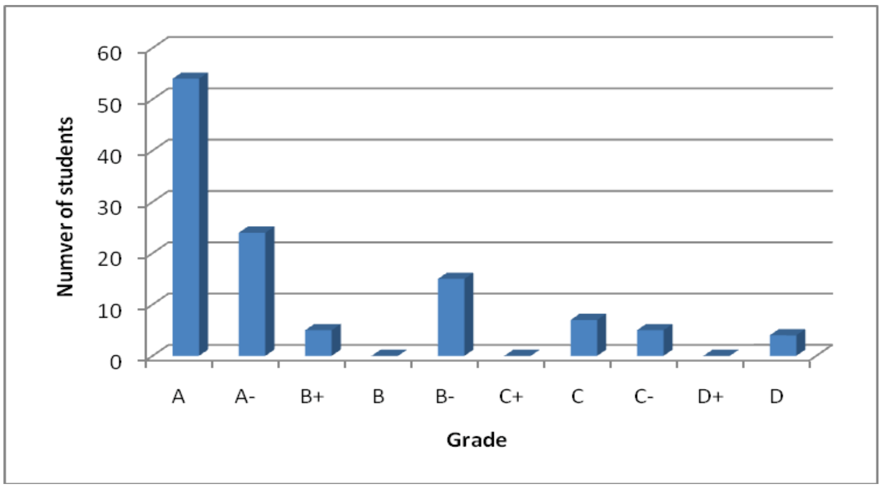

Fig 3. Students Grade for Workshop II in One Semester

5 THE CONCLUSION

This paper presents simulation education applied in Workshop II as part of the software engineering curriculum for undergraduate students at university level. In order to undertake Workshop II, students requires knowledge and skills which are gained from several subjects enrolled earlier in previous semesters. Workshop II plays an important role in the curriculum to allow application and practice of the knowledge and skills obtained in several essential subjects in the software engineering area.

These knowledge and skills are basically applied in Workshop II. Simulation education is seen essential in software engineering curriculum as it is not only a body of knowledge but more to competence. Through simulation, students are trained to deal with real problems in the right atmosphere. The effort eventually prepares the students to face a real working environment when they are graduated from the university.

REFERENCES

[1] M. B. Blake, *A student-enacted simulation approach to software engineering education*, Education, IEEE Transactions on, 46 (2003), pp. 124-132.

[2] P. J. Denning, *Computer Science: The Discipline*, *Encyclopedia of Computer Science*, Anthony Ralston & David Hemmindinger (eds.) 2000, 1999.

[3] A. Drappa and J. Ludewig, *Simulation in software engineering training*, *Software Engineering, 2000. Proceedings of the 2000 International Conference on*, 2000, pp. 199-208.

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