Using Game-oriented Projects for Teaching and Learning Software Engineering

Shaowen Qin and **Carl Mooney**
School of Computer Science, Engineering & Mathematics, Flinders University, Adelaide, Australia
shaowen.qin@flinders.edu.au, carl.mooney@flinders.edu.au

***Abstract**: Effective teaching and learning of software engineering fundamentals involves hands-on practice. For this reason, a substantial group project assignment is designed into every software engineering (SE) topic at Flinders University. The aim is to specify group projects that are realistic but reduced in scope. However the success of such assignments from both student background knowledge and more importantly student engagement points of view has been variable. To stimulate students' interest and improve their engagement, game-oriented projects have been considered as possible better choices for students. This idea was put into practice as a University Teaching and Learning Innovation project. This paper shares our experience in using game specification and design projects for teaching and learning SE concepts and methods. It presents the rational, the activities, students' feedback as well as our reflection of this effort. While confirming that using game-oriented projects is a promising choice, the paper also reveals some useful insights for further study in this area.*

Introduction

A university Software Engineering (SE) education curriculum typically covers fundamental knowledge and skills outlined in the SWEBOK (IEEE, 2004) to prepare students as software engineers or IT professionals. At Flinders University, we offer a series of 4 Software Engineering topics, covering the disciplines of requirement engineering, analysis and design, testing and maintenance, and software process and quality engineering. To make the learning environment as close to the real world software development environment as possible, a substantial group project is always included in the teaching and learning package of a SE topic. It is through this project that students not only gain practical experience with the processes involved in the various stages of the software development life-cycle, but also gain knowledge that helps them appreciate the issues that actually shape and motivate those processes. Informal survey and student feedback confirm that such a hands-on approach is one of the most preferred modes of learning.

Due to the limited time available (one semester), selection of the group projects has to take into consideration students' familiarity with the relevant real world situation to speed-up understanding of the requirements. The aim is to specify group projects that are realistic but reduced in scope. Taking the topic on Object-oriented Analysis and Design (SE2) for example, past project assignments had included a Student Assessment Record Management System, a Project Management system, and an Art Gallery Management system. However the success of such assignments, from both student background knowledge and more importantly student engagement points of view, has been variable.

With the objective of enhancing teaching and learning outcomes of Software Engineering course at Flinders in mind, game-oriented group projects were proposed as the major assignment of SE topics. We were also considering the possibility of offering a common project focus across the series of SE topics so that students could have a synergetic experience on how the different disciplines transition, overlap and impact on each other.

This effort was sponsored by the University as a Teaching and Learning Innovation project. This paper shares our experience as well as our students' opinions of the project.

Project Rational

Teaching Software Engineering through computer game development projects seems to be a sensible choice, but it is so far still a fairly new practice. There are limited text books available on software engineering and computer games, but their primary focus is on teaching computer game programming to future game developers instead of teaching software engineering concepts and methods to IT and Software Engineering Students in general (Flynt, 2005; Rucker, 2003). Nonetheless, such books do provide useful references on linking software engineering and computer games, and they show that game development projects are effective in engaging students' learning interest.

The scope and complexity of computer games are substantial with multi-user, networked, strategy games being at the high end. Gaming has always been a source leader for computer technology, for instance, it has driven developments in computer graphics and graphics cards. The requirements of computer games represent the requirements of most other modern software applications that involve state management, simulation, visualization and interaction.

The word "game" might still be perceived as "time consuming entertainment software" by many people. However, research on the psychological aspect of computer games has revealed that good games involve intricate learning experiences, and have a lot to teach us about how leaning is changing in the modern world (Gee, 2003). Game, in the context of this project, covers all software applications that make use of the motivating and engaging factors of gaming while achieving a teaching/learning goal. Simulation of real world scenarios for training and decision making is a typical example of such applications.

Game development not only involves all of the fundamental knowledge and skills required for software development, but also provides a platform for integration of knowledge from many other disciplines, such as mathematics, physics, mechanics, visual art and more importantly, innovation and creative thinking.

In his article "Game Development: Harder than You Think", game development consultant Jonathan Blow began with "the hardest part of making a game has always been the engineering." He later quoted: "Certainly, to write good engine code, you need to have a good grasp of software engineering. But also, there's a lot of domain-specific knowledge required." (Blow, 2004)

Game engineering can be a suitable project focus for the SE topics. As topic coordinators of Software Engineering courses, we see many advantages to integrate game development into teaching software engineering requirement analysis, design, coding, testing, quality assurance and project management principles and best practices. It is felt that using computer game development for SE group projects could further stimulate students' interest and maximize their leaning potential. Students will be given hands-on practices with game-oriented projects to keep them engaged in the subject matter and consequently enhance their learning. Most students have had exposure to games and would find game development exciting or at least interesting. In conjunction with this, game development promotes students' creative thinking; students can develop their own game idea with a specified game engine. This would also allow them to work on an idea that interests them the most, instead of having everything prescribed. Furthermore this shortens the time needed to clarify project specifications since students have the background and students own the requirement. They do not need to see "customers" to understand what functionalities or features the software must have, as they are their own customers. The academic staff members do not need to provide or simulate such customers. The projects are less "artificial".

The availability of open source game development platforms made the visualization of game ideas possible with minimal coding. For example, in a pilot investigation carried out for the topic on Object-oriented Analysis and Design in 2006, students found it quite easy to understand the project assignment that was to specify and then design a new game scenario with the "Greenfoot" framework ("Greenfoot," 2006), an open source software that can be used to create a wide range of visible interactive object worlds (scenarios) in a two-dimensional grid.

Project Activities

The project involved the following activities which address the preparation, implementation, monitoring, and evaluation aspects of the project:

1. Identification of suitable open source game development environments, and setting up a suitable environment as common starting point. We investigated low-cost commercial game engines and open source game development platforms in order to make the game-oriented teaching and learning sustainable. An open source platform, Panda3D ("Panda3D," 2007) was chosen. We developed a simple and rather primitive role play game prototype to get a realistic estimation of the time and effort involved to become familiar with the development environment, and the prototype was also used as a basis for revamping or enhancement. It should be noted that this could consume much effort if not controlled properly.
2. Identification of a few game themes. Ideally, the themes should be relevant to the research interest and strength of the teaching staff, such as game-aide education and rehabilitation. Role playing games (RPG) is a popular choice that is also versatile in fitting difference themes.
3. Reviewing teaching material to support the game development focus. Since our focus is not to prepare our students to be future game developers, rather to teach software engineering concepts and methods to future IT professionals and Software Engineers, we did not need to change a substantial amount of our existing teaching material.
4. Piloting game development projects in SE topics. For example, students doing Object-oriented Analysis and Design were asked to revamp and enhancing the Role Playing Game prototype.
5. Evaluation of the effectiveness of the project outcomes through survey of feedback from all relevant channels especially students, tutors and staff. A special purpose survey was conducted and its results are presented in the following section.

Outcomes

The survey questions were written with the clear intent of eliciting information regarding how the students felt about using game-oriented projects for learning the subject matter. However, as a result of this initial proposition, it was possible to ask about other scenarios (non- games) that may be applicable if the game scenario was not met favourably. Furthermore, we were also interested in the size of group that the students viewed as being optimal for this type of undertaking. With these notions in mind the following questions were formulated (see Table 1 for details).

Table 1 Questions used for the survey

1.	Do you believe that using the "Game" paradigm was a good engagement strategy for the group project work in Software Engineering 2 in 2007?
2.	Given the following choice of scenarios for the group project, please place them in order of preference: a. Online Magazine b. Art Gallery c. A Game d. Project Management Tool e. Student Management Assessment System
3.	Please write a brief statement (one or two sentences) describing why you ordered the list the way you did.
4.	Given that a project is to be a part of the assessment criteria, would you prefer to be given a written System Requirement Specification (SRS) from which to model the system, or would you prefer to reverse engineer an existing product (complete or not) to reach the model?
5.	Do you think it is possible to conduct this type of modelling without being in a group?
6.	If Not, what size group would you say is optimal a. 2 b. 3 c. 4 d. 5 e. 6
7.	Why do you think that size group is optimal?
8.	Should the group project be split into smaller manageable chunks for which shorter deadlines are imposed?

9. If so, what measures would you envisage would be of benefit?
10. Should all assessment be individual or should part of the assessment be a group component?
11. Would it be beneficial to carry a common theme or thread of project across all variants of the Software Engineering Suite of topic i.e. SE1, SE2, SE3 & SE4?
    a. If yes, would a Game scenario be a viable choice or are there alternatives?
    b. If no, do you have any suggestions that enable some continuity between the Topics
12. Would you be interested in carrying out some programming work to prototype your project in exchange for some extra marks (for example 10 %)?
13. If you could include a work experience component into the assessment methods, where would you expect the marks to be transferred from?
    a. i.e. The work experience component is worth 5%, would you then make the exam, group project phase 1, or group project phase 2 worth 5% less?
14. As a final question is there one type of approach or scenario that would engage you more than any other? If so what would that be?

The survey was conducted on-line using the Flinders Learning On-Line (FLO) educational package and was available for a one week period. We felt that it was necessary to allow a longer period of time – longer than a tutorial session – so that the results would not be of "the first thing that comes into my head" type answers, but more considered given the allowable time. 55 students responded to the survey. The results, as seen in Figures 1 to 8, are discussed briefly as follows.

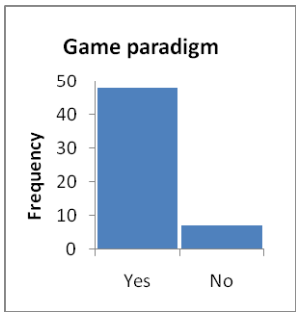

Figure 1: The number of respondents who believed that an RPG scenario was a good choice.

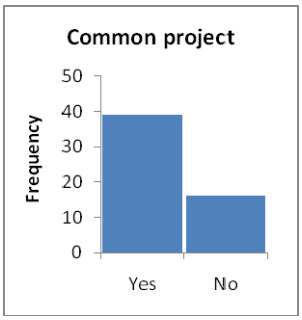

Figure 2: Should a common project be carried through the entire Software Engineering stream?

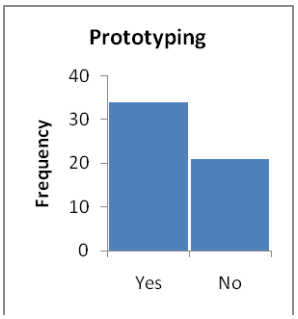

Figure 3: Should prototyping be an included as part of the assessment?

Figure 1 shows that, for the question of whether a Role Playing Game (RPG) scenario was considered a good paradigm for learning the concepts of object-oriented analysis and design, 87.3% of respondent agreed whereas only 12.7% disagreed. This is viewed as a very positive response to a game-oriented project, as we understand and acknowledge the fact that not everyone likes games.

Figure 2 shows the response to the question of a common thread or project to be carried through all of the Software Engineering offerings. It indicated that 70% were in favour and 30% against. This is not surprising since it would reduce the cognitive load on students from year to year and would allow for a reasonable if not fully working model of the project selected. The problem for academics if this were to be implemented is that a 'fall-back' position would have to be implemented for those students whose work was not up to standard or who were new to the course.

The closeness of the counts for the inclusion of prototyping in Figure 3, 62% for and 38% against, could be due to a number of factors, the first, a lack of programming skills on behalf of the respondents and the second the amount of work that it would entail.

As seen in Figure 4, the choice of a game scenario for the current course differs from the result seen in Figure 1 if other types of scenarios are able to be selected – 53% for a game and 47% for a grouping of other scenarios. This could be seen as a reflection that on average half of the students are interested in games and the other half have other specific interests. If we take into account the second choice outcomes shown in Figure 5 for those students who did not select game as the first choice then a game scenario still ranks high at 35%, which equates to 16.5% of the overall cohort. This addresses

the gap between Figure 4 and **Error! Reference source not found.**. Across the board however (see Figure 6) as a second choice students tended to lean towards scenarios that they have some underlying knowledge or exposure to, as evidenced by the 36.4% response to an On-line Magazine.

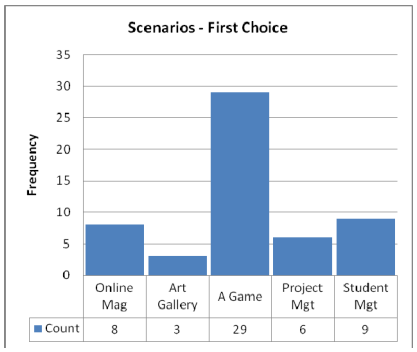

Figure 4: First choice selections for the type of project to be undertaken

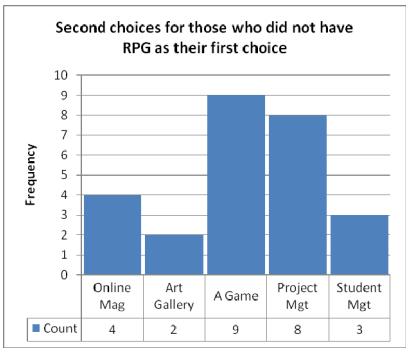

Figure 5: Second choice selection for the type of project to be undertaken from those who did not choose a RPG as a first choice

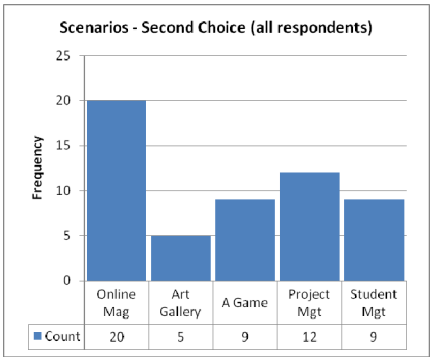

Figure 6: Second choices for all respondents

Figures 7 & 8 are included to highlight the choices in optimal group sizes and the different scenarios discussed above and indicate that group sizes of 4 and 5 are favoured regardless of the intention of a common theme and whether prototyping is included or not. An interesting observation is that smaller groups are favoured in the non-RPG scenarios.

Conclusion and Future Work

An experience report of using game-oriented project for software engineering group assignment at Flinders University is presented. Student feedback, after having actually worked on a game-oriented project, confirmed our expectation of improved engagement, which should contribute to a better learning outcome. We did not attempt to measure the amount of improvement introduced by the game project at this stage, given our limited experiences and the many variables that would influence the outcome, but we would certainly be looking into measuring student performance improvement as more data becomes available. We would also like to gauge how well received a game-oriented project option would be after the students had conducted a non-game type project. As stated earlier, 36% of the students opted for an On-line Magazine project as their second choice and as such, this has been adopted as the scenario for a follow up study. We plan to compare students' preferences with and without a game project experience to further exam the objectiveness of our current conclusion.

Game-oriented software engineering education is forecast as one of the future trends of software engineering (Boehm, 2006), and there is much work to be done to bring its potential educational value into fruition. Efforts along this line are continuing at Flinders University. For example, a multi-media PC lab with game development environment is being set up to support game development

projects. On the other hand, student feedback did indicate that not all students like games, and the real challenge is to be able to meet the differing needs of all students.

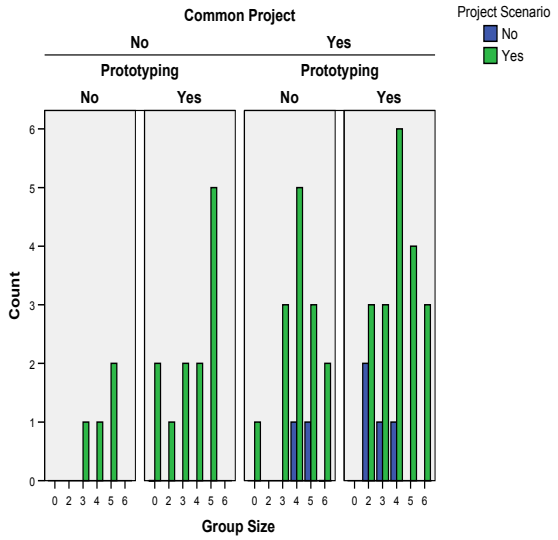

Figure 7: Clustered bar chart of those choosing a Common Project and Prototyping mapped against the optimal number of Group Members indicating their preference for an RPG scenario.

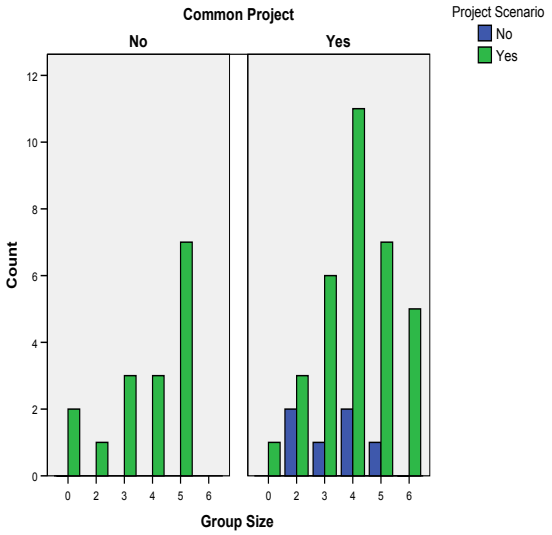

Figure 8: A Common project theme across all Software Engineering offerings mapped against preferred group size with the indication of an RPG or not.

References

Blow, J. (2004). Game Development: Harder Than You Think, ACM Queue.

Boehm, B. (2006). A view of 20th and 21st century software engineering. Paper presented at the Proceeding of the 28th international conference on Software engineering

Flynt, J. P. (2005). Software Engineering for Game developers: Thomson Course Technology

Gee, J. P. (2003). What video games have to teach us about learning and literacy: Palgrave Macmillan.

Greenfoot. (2006). (2006).

IEEE. (2004). Guide to the Software Engineering Body of Knowledge (SWEBOK).

Panda3D. (2007). (2007).

Rucker, R. (2003). Software Engineering and Computer Games: Addison Wesley.

Acknowledgements

The Authors wish to acknowledge the support from Flinders Teaching and Learning Innovation Fund for this work.

Copyright © 2009 Remains the property of the author(s). The author(s) assign to AaeE and educational non-profit institutions a non-exclusive licence to use this document for personal use and in courses of instruction provided that the article is used in full and this copyright statement is reproduced. The author(s) also grant a non-exclusive licence to AaeE to publish this document in full on the World Wide Web (prime sites and mirrors) on electronic storage and in printed form within the AaeE 2009 conference proceedings. Any other usage is prohibited without the express permission of the author(s).