Role-Based Meets Problem-Based for Software Engineering Learning

Thanwadee Sunetnanta[1], Olly Gotel[2], Vidya Kulkarni[3], Moniphal Say[4], Christelle Scharff[2]

[1]Mahidol University, Bangkok, Thailand, cctth@mahidol.ac.th
[2]Pace University, New York, USA, ogotel@pace.edu, cscharff@pace.edu
[3]University of Delhi, Delhi, India, vkulkarni@cs.du.ac.in
[4]Institute of Technology of Cambodia, Phnom Penh, Cambodia, say.moniphal@gmail.com

Abstract

Problem-based learning (PBL) has been proving its success in many areas of education that require experimental learning or 'learning by doing', especially in engineering, medical, dental and veterinary education. In a typical PBL environment, students are assigned a real world problem to explore and then work collaboratively in groups to seek solutions to that problem. In PBL, problems are often tackled from a single perspective, e.g., from the perspective of a problem-solver. Therefore, less emphasis is given to the development of some other skills, such as communication skills, management skills, change management skills, negotiation skills, risk management skills, quality management skills, etc. These skills are, however, rather crucial in the context of the globalization of software development where the differences in culture and language must be resolved amongst a variety of roles and perspectives to reach a shared goal.

This paper presents our view on evolving PBL through its convergence with role-based learning. In particular, the paper shares our experiences from collaborating on a Global Software Development (GSD) project in 2008, a project that involved students from multiple universities in four different countries. Different roles and responsibilities were established in this GSD project for the different competencies that were needed on the project to nurture those skills desirable for working in global settings. Particular roles like mentoring and auditing were also set up to strive for quality throughout the duration of the project. The paper considers the talent and skills that can be cumulatively acquired through a specific role assigned in such a project. Reflections on adopting this combination of role-based and problem-based learning were then applied to a local software engineering class in Thailand. This experience is highlighted to point out the transition in the learning itself and the steps required to tailor a local class setting to gain some of the same benefits of such a model of global work.

1. Introduction

Software engineering is the study of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software [1]. The goal of software engineering is towards the quality of software development in terms of both product quality and process quality. The study of software engineering draws its foundation from the emergence of a variety of disciplines including the computing discipline, engineering discipline and professional practice [2]. In an educational context, software engineering teaching and learning are therefore the combination of arts and sciences. Software engineering students are required not only to master subjects related to computing and engineering techniques for producing high-quality software, but also to attain software engineering specialist knowledge with appropriate professionalism, including work ethics, teamwork, communication, management and analytical skills.

Notably, software engineering learning is action-oriented [3]. Typical software engineering related-courses require students to work on capstone projects in order to employ knowledge gained from the courses for the development of a software system. Such a capstone project in software engineering education can be considered as a form of small group problem-based learning (PBL), which has been proving its success in many areas of education which requires experimental learning or ‘learning by doing’. Examples of specific areas of study which often use PBL are engineering, medical, dental and veterinary education. In a conventional PBL environment, students are assigned a real world problem to explore and they work collaboratively in groups to seek solutions to that problem.

Some previous attempts in applying PBL to software engineering are presented in [4] [6] [7]. In some cases, PBL in software engineering is also deployed within a physical environment similar to what students will experience in their future workplace [8]. Similar works also aligned with PBL practice in the study of computing and software engineering are studio-based approaches [9] [10]. Instead of exercising PBL in a single course, studio-based approaches integrate aspects of real-life software development projects and computing practice throughout the duration of a degree and, in different levels or years of study, gradually expose students to the issues, situations and scenarios they will potentially encounter in the work environment.

Our work subscribes to the idea of the previous works in applying PBL to software engineering as mentioned but adds another dimension of role-based practice into PBL exercises. In PBL, problems are often tackled from a single perspective, e.g., as a problem-solver. Although PBL also focuses on achieving teamwork to a certain degree, less emphasis is given on the development of other skills required for software engineering professional practice, such as communication skills, management skills, change management skills, negotiation skills, risk management skills, quality management skills, etc. With the advent of offshore and outsourcing software development, these skills are rather crucial in the context of globalization where the differences in culture and language must be resolved amongst a variety of roles and perspectives to reach a shared goal.

In view of the demands of a global setting, this paper presents an evolution of PBL through its convergence with role-based learning. This paper, in particular, is inspired by the discussion in Delaney et al. [4]. However, we propose to extend that of “Software Engineering Meets Problem-Based Learning” to “Problem-Based Learning Meets Role-Based Learning”, thereby enabling the transition of both problem-based learning and role-based learning to the software engineering area. A similar application of role-based learning in software engineering is found in [5]. Unlike the work in [5], which deploys role play in computer games for teaching software engineering, we encourage students to practice role play physically, not virtually.

In this paper, we share our experiences from our collaboration on a Global Software Development (GSD) project in 2008, which involved students from multiple universities in four different countries. Section 2 of this paper describes the setting of the GSD project and our experiences with the 2008 GSD in detail. This collaboration led to the adoption of role-based and problem-based learning in a local Thai software engineering class, and this experience is described in Section 3. We also highlight the talent and skills that can be cumulatively developed through a specific role assigned in a project. Section 4 reflects upon adopting both role-based and problem-based learning in combination to point out the transition in the learning itself that results and the steps required to set up a class following such a model.

2. The Collaboration of the 2008 Global Software Development Project (GSD)

The Global Software Development (GSD) project is an innovative research project on software engineering teaching and practice through a challenging global collaboration. The website for this collaboration in 2008 can be found at http://atlantis.seidenberg.pace.edu/wiki/gsd2008. The GSD initiative was started by Pace University in New York City and the Institute of Technology of Cambodia (ITC) in Phnom Penh in 2004 to examine different models of working to enable students to collaborate and develop software together effectively whilst addressing institution-specific learning needs and accounting for technological constraints. The collaboration was scaled up throughout the years to link students from more universities in different countries and to bring together undergraduate, graduate and industry students to explore different roles in software development. The evolution of this GSD project is reported in [11] [12] [13].

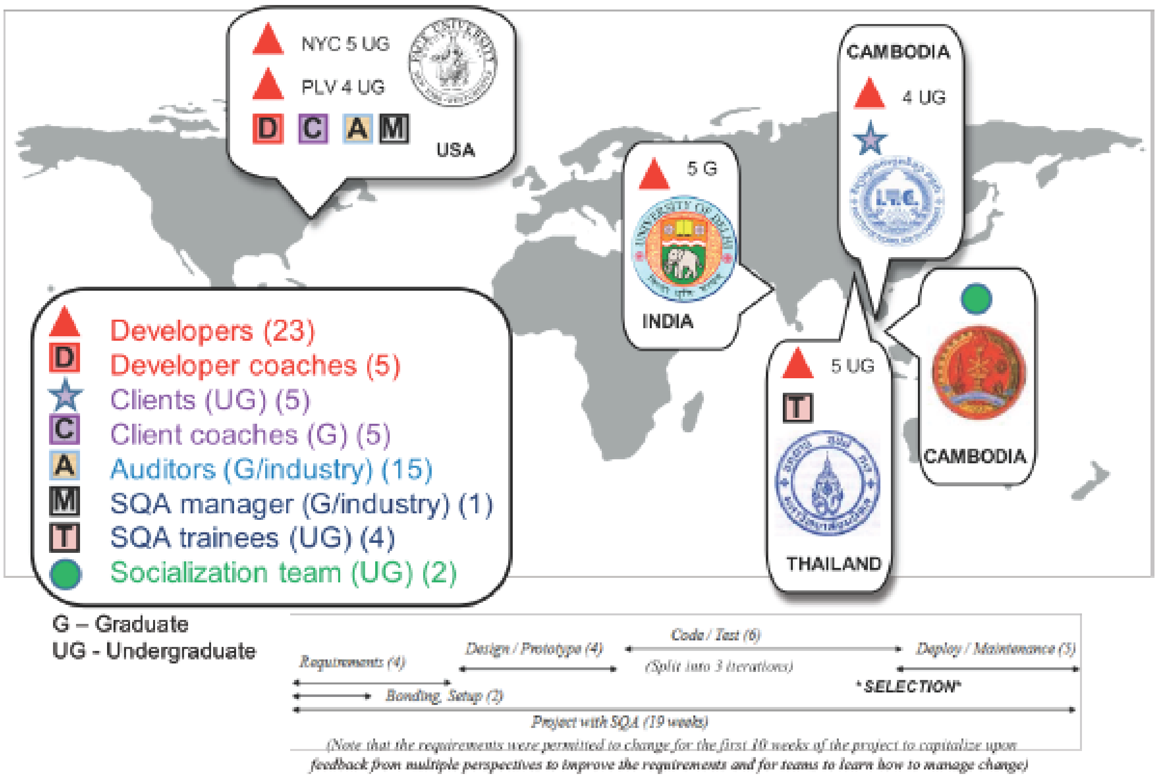
NYC 5 UG
PLV 4 UG
D C A M
USA
CAMBODIA
4 UG
5 G
INDIA
5 UG
T
THAILAND
CAMBODIA
Developers (23)
Developer coaches (5)
Clients (UG) (5)
Client coaches (G) (5)
Auditors (G/industry) (15)
SQA manager (G/industry) (1)
SQA trainees (UG) (4)
Socialization team (UG) (2)
G – Graduate
UG - Undergraduate
Requirements (4)
Design / Prototype (4)
Code / Test (6)
(Split into 3 iterations)
Deploy / Maintenance (5)
Bonding, Setup (2)
SELECTION
Project with SQA (19 weeks)
(Note that the requirements were permitted to change for the first 10 weeks of the project to capitalize upon feedback from multiple perspectives to improve the requirements and for teams to learn how to manage change)

Figure 1. The GSD Model in 2008 (Reproduced from [12])

In 2008, the GSD project spanned four countries, five universities and three time-zones, and lasted for fifteen weeks (from January 2008 to May 2008). Teams of student developers from Cambodia, India, New York City, New York Pleasantville and Thailand competed to design and develop a software system for selection. The problem theme in year 2008 was MultiLIB 2008, an electronic library system for the Computer Science Department of the Institute of Technology of Cambodia. The project also involved a team of student clients in Cambodia responsible for determining the school’s requirements and managing the entire procurement process; they worked directly as sponsors for the five global development teams. Each team worked with a set of dedicated coaches and auditors, graduate software engineering students and IT professionals from New York, who both mentored and focused on the quality of the processes and products.

The GSD project resulted in five excellent software systems for the clients to consider and all were delivered to a strict deadline. The final project concluded after the clients, having reviewed all five systems, identified each one’s strengths and weaknesses, then selected the system that stood out as the most complete and thorough electronic library system for further deployment. The students from Cambodia, India, New York City and New York Pleasantville joined this 2008 GSD project as part of their software development, database or software engineering courses. Due to the difference in semester system, the junior and sophomore Thai students from the Information and Communication Technology (ICT) Programme, Computer Science Department, Faculty of Science, Mahidol University joined this GSD project during their semester break as part of their internship programme. Figure 1 is the GSD model in 2008 which is reproduced from [12].

As seen in Figure 1, the GSD project setting for 2008 strives for quality in collaborative software development through role-based learning of six main types of software development stakeholder, i.e., clients, client coaches, developers, developer coaches, Software Quality Assurance (SQA) manager and quality auditors. Socialization teams and SQA trainees are not considered as the main roles since they played secondary supporting activities on the project. The socialization team was responsible for increasing socialization with respect the client/developer relations in order to make bonding amongst the students from different universities an extra-curricular activity so as to help resolve any cultural differences that might be encountered. SQA trainees who were undergraduate students without much knowledge in software quality assurance played a shadow role to observe the role of quality coaches and quality auditors, and learned the value of quality by working with the Thai coaches and auditors.

Table 1 summarizes the responsibilities, skills and competencies to practice for the main roles of the 2008 GSD project. Figure 2 depicts the overlapping of skills and competencies of different roles. Since the GSD project aims at nurturing a collaborative development environment, we can see that the common skills and competencies required for every role in the project are the interpersonal and communication skills. Feasibility study and project selection processes are emphasized for the roles of clients and client coaches, while the design, programming and change management skills are emphasized for the roles of developers and developer coaches. Nonetheless, clients, client coaches, developers and developer coaches share the need to practice analytical skills and negotiation skills in order to solve the problems encountered and produce the required software products. The quality of the required software products is driven through the practice of quality management between the developers, developer coaches, quality auditors and SQA manager. The quality auditors and SQA manager also take an important role in practicing risk management to ensure that a product with requisite quality will be delivered at the end of the project by conducting project health monitoring and tracking activities. Clients and client coaches share the view of the project management with quality auditors and the SQA manager so as to practice more on process improvement. Client coaches, in particular, exercise the risk management activities with the quality auditors. The sets of skills and competencies that are represented as nested circles, i.e., conflict management, are those that are specifically required for the coaching and mentoring roles of client coaches, developer coaches and the SQA manager.

Table 1. Responsibilities and Competencies of the Main Roles Expected in the GSD Model

Role	Responsibilities	Practice of Skills and Competencies
Clients	- Own and manage requirements - Solicit candidate software systems to meet the requirements - Conduct acceptance test - Select the systems that of highest quality to further deploy	- Analytical skills for problem solving - Negotiation - Project management - Feasibility study and project selection process - Interpersonal and communication
Client Coaches	- Mentor the client teams to baseline the requirements and manage changes - Formulate test plans - Guide selection process	- Analytical skills for problem solving - Negotiation - Project management - Feasibility study and project selection process - Risk management - Conflict management - Interpersonal and communication
Developers	- Develop a software system to satisfy the requirements	- Analytical skills for problem solving - Negotiation - Design - Programming - Change management - Interpersonal and communication
Developer Coaches	- Mentor developers with project planning, analysis, design and implementation activities - Conduct code walkthrough to inspect coding problems - Review auditor comments and suggest to the developers how to improve the developed system - Review test cases and testability matrix diagram	- Analytical skills for problem solving - Negotiation - Change management - Conflict management - Quality management - Interpersonal and communication
SQA Manager	- Facilitate auditing process of quality auditors - Track and report project health check from feedback obtained from quality auditors	- Project management - Quality management - Risk management - Conflict management - Interpersonal and communication
Quality Auditors	- Independently check the processes and products of each development team - Provide feedback to assure and improve quality - Review the requirements document at regular periods	- Project management - Quality management - Risk management - Interpersonal and communication

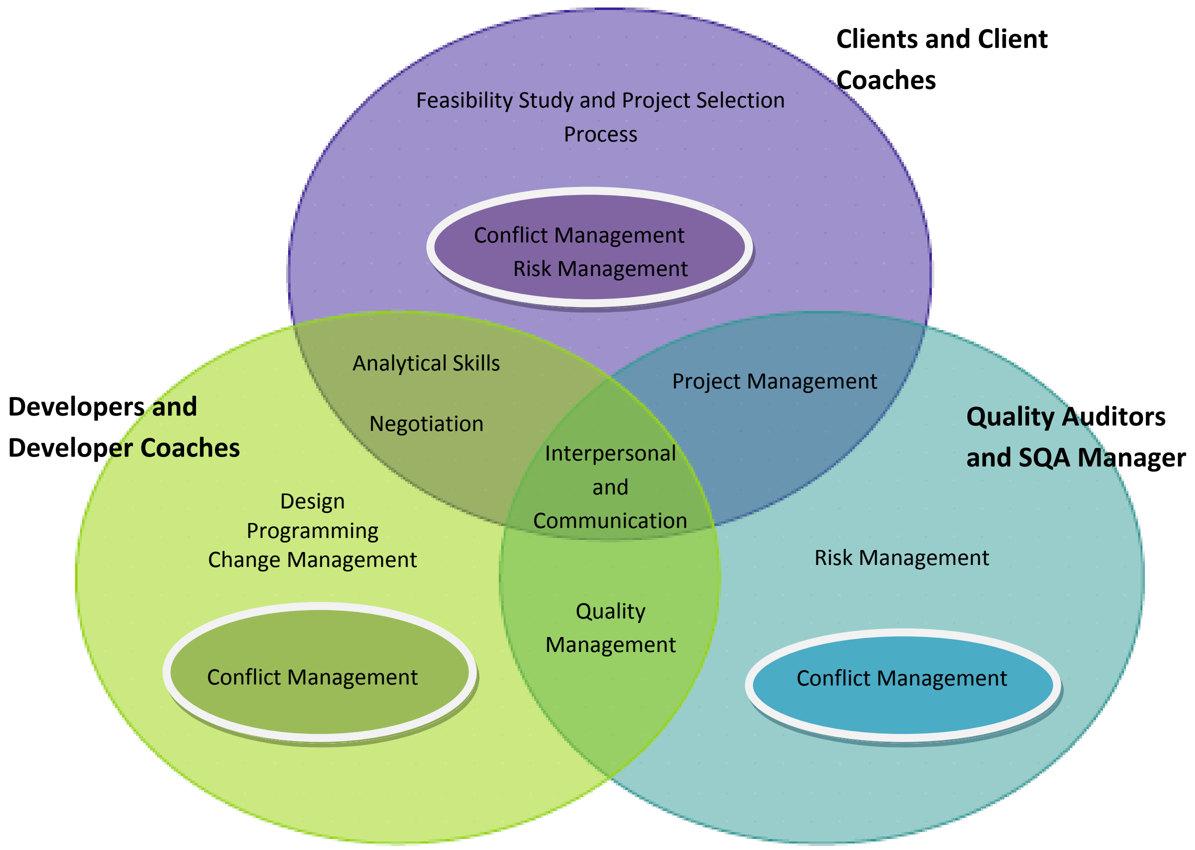

Figure 2. Skills and Competencies to Practice for Different Roles of PBL in Software Engineering

3. Role-Based Meets Problem-Based in Thai Software Engineering Classes: Local Adoption of GSD

In the semester following the period of the 2008 GSD project, the Thai instructor who joined the project has adopted the GSD setting to teach a course on software analysis and design at the Information and Communication Technology programme, Mahidol University. The adoption aims at reproducing the teaching and learning experiences gained from the GSD project to better provide a learning-centered environment for students to encourage the students to generate ideas for better practice in software engineering. The course was started in June 2008, ran for sixteen weeks and was taken by 208 students (3 class sections). Students were divided into groups of 5-6 students and 23 of these groups (from 2 class sections) did their projects based on this GSD setting. The majority of the participating students were in the third year of the programme. The course introduced the students to the principles of software engineering, software process and related modeling techniques for software analysis and design. The projects that the students undertook under this GSD model were varied depending on the subjects or problem domains of interest of the students who played the client role. The students were asked to pretend to be an owner of a business of their interest. Some examples were a homemade bakery, a bouquet ordering service, and resort reservations.

As part of the course's evaluation, students were required to undertake term-projects in the same way that typical PBL is conducted for software engineering classes. Students were grouped into project teams, each comprising five or six members. Different teams of students were given different problems to which they were required to develop a software system as a solution. Project deliverables were produced in a timeline manner that corresponded to the different phases in software development. The deliverables included the scope of the work, use case models, use case specifications, process models, data models and prototypes. The due dates of the deliverables were aligned with the content of the course. For example, students were required to submit the scope of the project once they finished the classes on project initiation and requirements gathering. Similarly, students were required to submit use case models and use case specifications once they finished the classes on modeling system requirements using use cases, and to submit process models and data models once they finished the classes on process analysis and data analysis respectively. Initial prototypes from each project team were required two weeks before the course ended to verify the design of the system interfaces. Every team was asked to submit their final prototypes together with a complete set of project documents containing the details of all the deliverables to cross check the transition of analysis and design results to the implementation of the prototypes.

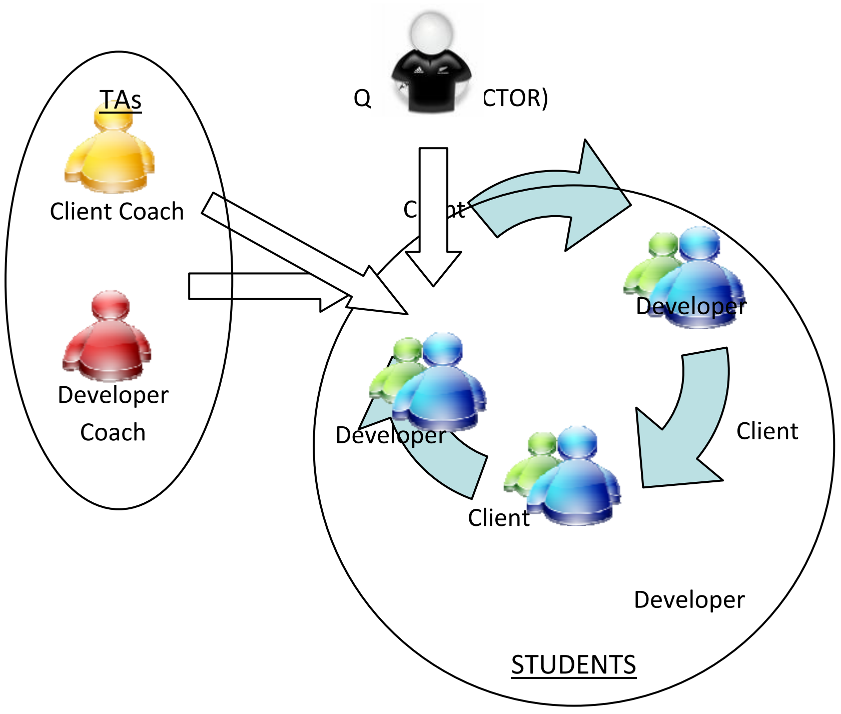

Figure 3. The Adoption of the GSD Role-Based Model to a Local Software Engineering Class in Thailand

The adoption of the GSD model to this course has some constraints and limitations. First, the adoption was for the undergraduate students only and the students had very little prior understanding of software development processes. This is because the course is the first introductory course to the software engineering area of study. Secondly, it was set as the course objective that the students were required to not only learn about the software development process and modeling techniques, but also to develop their programming skills through their term projects. Accordingly, to deploy the role-based learning as in the GSD project to the course, a single group of students that teamed up played two roles at the same time. Instead of having the course instructors assign the problems to the students to solve, students were asked to come up with the problems according to their interest. However, unlike some typical software development courses, students were not asked to solve the problems that they came up with by themselves. In place of that, they were asked to develop a system based on the problem given by another group. This was done intentionally so that a group of students can attempt to practice skills and competencies from both perspectives of clients and developers. Figure 3 shows how the roles were assigned to the groups of students.

From Figure 3, we see that the teaching assistants (TAs) for the course played the roles of client coaches and developer coaches. At the time the course was conducted, there were two teaching assistants for the course. This is different from the original GSD model because there was no graduate student that could take the coaching role as described earlier. Also, the two course instructors carried out the roles of quality manager and quality auditor for similar reasons. The TA coaching and mentoring responsibilities were to give guidance to the clients and developers to clarify requirements and help conduct the various activities required. In some cases, when acting as developer coaches, the TAs also helped to resolve technical difficulties in the project implementation.

4. Lesson Learned and Suggestions

The practice of role-based learning as employed in the GSD project has significantly improved the problem-based learning environment in the course described in Section 3. From the instructor observation, the students took a shorter learning curve in the problem analysis phase, primarily as they were more enthusiastic to do the projects to satisfy the requirements from the clients, who are also their friends. Unlike typical software engineering capstone projects, where students often come up with the problems by themselves, it was found that the combination of problem-based learning and role-based learning made the students more motivated to deliver the software solutions to the problems that were initiated or given by someone else. Also, it was less demanding on the teaching assistants and the instructors of the course to clarify the requirements the students obtained since those who played the client roles did that job as part of their responsibilities. The clients also enjoyed commenting on the resulting work products from the developers, who are also their friends. From a teaching point of view, the model does create a new learning environment that encourages collaboration. Moreover, role-playing is a very suitable method for the students to explore the issues involved in complex social situations when they have to look from the different perspectives of both customers and developers at the same time. The model also enabled the assimilation and practice of a broader set of skills and competencies, as required for different roles in software development, as well as nurtured those skills desirable for working in global settings.

The adaptation of the GSD model presented in this work is different from the general ideas of employing a mentoring role in the PBL environment in the way that the role of mentors in the model focuses on driving the quality of the work rather than guiding the students to the solutions of the problems as some other PBL-adopted classes might focus attention. The solution mainly resulted from the collaboration and negotiation between those who played client roles and those who played developer roles.

The difficult part in adopting the GSD model to the course was in the planning of the supporting environment and tools. Since the instructors had minimal time to plan for this course right after the GSD project itself ended in May 2008, the tool that was adopted from the GSD project was only to facilitate the communication through wikis. The wikis were also used as an observation tool for the course's instructors to periodically monitor and inspect the work items and work products of each project group to provide feedback to assure and improve the quality of the work produced. Likewise, the teaching assistants who played the coaching roles could refer to the information uploaded onto the wikis for mentoring purposes.

The better the plan and preparation of the supporting environment and tools for the adoption of the GSD model would no doubt make the model itself more efficiently exploited.

The following are the steps suggested to tailor a local class setting to gain some of the similar benefits of such a model of global work as we have experienced:

- Step 1: Prepare project plan and line up expected deliverables.
- Sept 2: Prepare computer supported collaborative work environment and tools.
- Step 3: Let the students team up.
- Step 4: Arrange project kick off to declare roles and responsibilities of different teams.
- Step 5: Create social networks among different teams to share a goal by using electronic communication tools, such as a wiki, to facilitate collaborative work.
- Step 5: Encourage the students to do their work through collaboration.
- Step 6: Monitor the work and interaction among different teams as the project progresses and mentor when necessary. This can be done by periodically checking the work products that are uploaded to the wiki or the deliverables to track project progress. In addition, students may also be asked to upload their minutes of meetings so that the frequency of the communications can be monitored.
- Step 7: Conclude and evaluate the results of the project and skills learnt for different roles.

Another area of improvement would be to accommodate other software process models into the learning environment. For the sake of simplicity and as the first step of adopting and practicing the GSD model locally, we asked the students to undertake their software development activities in a sequential order following the waterfall process model. Hence, the project started with project initiation, followed by activities required for system analysis, system design and system implementation. A study on adopting other pedagogical methods, which will encourage students to learn more about software process models, such as the work in [14] about agile process models, is currently being studied and investigated further.

5. References

[1] IEEE STD 610.12-1990, *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Computer Society, 1990.

[2] Software Engineering 2004: Curriculum Guideline for Undergraduate Degree Programs in Software Engineering, IEEE Computer Society, 2004.

[3] M. Murphy, and R. Hastead-Nussloch, “Learning and Instructional Issues in Software Engineering”, Workshop: Beg, Borrow or Steal: Using Multi-disciplinary Approaches in Empirical Software Engineering Research (ICSE’2000), Limerick, Ireland, June 4-11, 2000.

[4] J. D. Delaney, G. G. Mitchell, and S. Delaney, “Software Engineering Meets Problem-Based Learning”, The Engineers Journal, 57 (6), Institution of Engineers of Ireland, 2003.

[5] T. M. Connolly, M. Stansfield, and T. Hainey, “Using Games-Based Learning to Teach Software Engineering”, Lecture Notes in Business Information Processing, (LNBIP), Vol. 8, pp. 304–313, 2008.

[6] L. Brodie, H. Zhou and A. Gibbons, “Steps in Developing an Advanced Software Engineering Course Using Problem Based Learning”, Engineering Education Journal, Vol. 3, Issue 1, 2008, pp. 2 - 12.

[7] J. Ryoo, F. Fonesca, and D. S. Janzen, “Teaching Object-Oriented Software Engineering Through Problem-Based Learning in the Context of Game Design”, In Proceedings of the 21st Conference on Software Engineering Education and Training - Charleston, SC., April. 2008.

[8] Sita Ramakrishnan, “MUSE Studio Lab and Innovative Software Engineering CapStone Project Experience”, In Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education, ACM SIGCSE Bulletin, Volume 35, Issue 3, September 2003, pp. 21 - 25.

[9] M. Simpson, J. Burmeister, A. Boykiw and Jihan Zhu, “Successful Studio-Based Real-World Projects in IT Education”, The Australasian Computing Education Conference (ACE2003), Conferences in Research and Practice in Information Technology, Vol. 20, Tony Greening and Raymond Lister (Eds.), 2003.

[10] C. D. Hundhausen, N. H. Narayanan, and M. E. Crosby, “Exploring Studio-Based Instructional Models for Computing Education”, SIGCSE’08, March 12–15, Portland, Oregon, USA, 2008.

[11] O. Gotel, V. Kulkarni, M. Say, C. Scharff, T. Sunetnanta, S. Touch, and P. Des, “Quality-Driven Competition: Uniting Undergraduates, Graduates and Professionals on Global Software Development Projects”, Workshop on the Roles of Student Projects and Work Experience in Undergraduate and Taught Postgraduate Programmes, The 21st Conference on Software Engineering Education and Training (CSEET 2008), Charleston, South Carolina, USA, April 14-17, 2008.

[12] O. Gotel, V. Kulkarni, M. Say, C. Scharff, and T. Sunetnanta, “A Global and Competition-based Model for Fostering Technical and Soft Skills in Software Engineering Education”, In Proceedings of the 22nd IEEE-CS Conference on Software Engineering Education and Training (CSEE&T'09). Hyderabad, India, 17-19 February, 2009.

[13] O. Gotel, V. Kulkarni, M. Say, C. Scharff, and T. Sunetnanta, “Evolving an Infrastructure for Student Global Software Development Projects: Lessons for Industry”, In Proceedings of the 2nd India Software Engineering Conference (ISEC’09). Pune, India, 23-26 February, 2009.

[14] M. Isable, and A. Botia, “An Iterative and Agile Process Model for Teaching Software Engineering”, In Proceedings of the 18th Conference on Software Engineering Education & Training (CSEET’04), 2005, pp. 9 - 16.

Acknowledgement

The first author would like to express special thanks to Associate Professor Dr. Jarernsri Mitrapanont, who was the director of Mahidol University Computing Center at the time that the Thai team participated in the 2008 GSD project and who is now the acting dean of the Faculty of Information and Communication Technology at the time this paper is being written, for providing all the facilities required for undertaking this project and for her encouragement and support on joining the GSD project.