Reflection and guided problem-based learning in software engineering

Ljiljana Brankovic, Huilin Ye and Michael Reynolds
School of Electrical Engineering and Computer Science, The University of Newcastle
Corresponding Author Email: *Ljiljana Brankovic@newcastle.edu.au*

Structured abstract

BACKGROUND
The context of this study is a Software Engineering Project, taught in the 3rd year of BEng(Software) at the University of Newcastle in 2011/2012. This is a two-semester course where the students work in teams of 4-5 members to develop a real-life software application – FlightPub, a website where customers can search for and book airline flights, as well as become members and collect loyalty points. To succeed in such a challenging project, students need to apply all the knowledge assimilated in their previous years, as well as to acquire other knowledge and practical skills necessary for successful completion of the project. Contact hours consisted of a two hour lecture, a two hour workshop and a 30minute mentor meeting for each of the teams separately, per week.

PURPOSE
The Software Engineering Project is a challenging course both for students and teaching staff, as the tasks are open ended and teams are required to work more independently then in any of their previous courses, and thus new teaching approaches needed to be explored and/or developed, that would better facilitate student learning.

DESIGN/METHOD
To support the students' needs, we developed 'guided problem based learning' and adopted reflective learning approaches, both of which were assessed individually, rather than as a team. Guided problem based learning was utilised mostly for exploring and adopting software tools, such as version control, software frameworks, and project management software, whereas reflective learning was assessed in the form of a reflective journal within the final report. We evaluate the effectiveness of the teaching methods based on the overall success of the software performance, the extent to which software tools were used in the project, as well as formal and informal feedback from the students.

RESULTS
Guided problem based learning is an effective learning approach in project based software engineering, while reflection is extremely valuable in improving the problem solving and decision making skills of students involved in the software development process.

CONCLUSIONS
Problem based learning naturally lends itself to software engineering projects, as the initial requirements are somewhat ill defined and all learning is in the service of accomplishing the job. However, pure problem based learning may not be suitable for software engineering projects, as students may find themselves overwhelmed by the complexity of the project and the diversity of possible directions, and thus guided problem based learning appears as a valuable alternative approach.

KEYWORDS
Software engineering project; problem-based learning; reflective learning

Introduction

It is generally accepted that Software Engineering and other computing degrees are well served with a capstone project course, and that industry based project courses are preferable as no lab designed project can ever adequately reflect veracity of a real life software development (Reichlmayr, 2006). The second author of this paper had been running such a course for over 10 years, with support of industry partner Object Consulting, one of the most reputable Sydney IT firms (Ye, 2009).

In 2011, this course was restructured as a 2 semester Problem Based Learning course, with an emphasis on Reflective Learning. In semester 2 2011 we introduced guided problem based learning into this course, and in 2012 we further refined our PBL and Reflective approach. In this paper we present this journey, as well as student perception on the course and their own learning.

The paper is organised as follows. In the next section we present previous work in the area of Project Based Learning (PrBL), Problem Based Learning (PBL) and Reflective Learning, and in the subsequent section we give an account of using Guided Problem Based Learning and Reflection in SENG3150/3160 at the University of Newcastle in 2011 and semester 1 2012. In the following section we present an evaluation based on student perception of the course and their own learning and in the last section we present some concluding remarks.

Previous Work

Project based learning in software engineering

Many IT, Computer Science and Software Engineering degrees include a capstone project course in their 3rd or 4th year. A capstone course is indeed one of the accreditation requirements of the Australian Computer Society (ACS/NZCS/ANZICT, 2009) and a project course appears to fit in perfectly: Students use and integrate much of the knowledge they acquired over the previous years, further develop and integrate skills and achieve a depth of ICT study (Graham and Johnson, 2012).

However, it is well recognised that even the best designed academic project cannot adequately prepare software engineering students for work in industry as it fails to capture dynamics of building a real world application (Reichlmayr, 2006). Therefore, project courses often have an industrial partner involved in some aspects of the project (e.g., Ye, 2009), which can range from a modest contribution, such as a being a client for the software, to approaching the student project as their own in-house development, treating students as their own employees and even using their own software development team to provide extensive testing for the student project (Johns-Boast and Patch, 2010).

Projects involving industry partners have a potential to be very successful and beneficial for all stakeholders, including students, industry partners and academics involved. Students have an opportunity not only to consolidate knowledge acquired at the university on a real-life project but also to gain work experience within industry, industry partners have an opportunity for community engagement, as well as access to potential employees and a chance to assess them beforehand, while academics can offer more realistic and engaging projects and foster relationships with industry (see Perrenet, Bouhuijs and Smits, 2000, and Johns-Boast and Patch, 2010, for more discussion on benefits and issues in industry based student projects).

However, despite all the benefits of the project-based learning, students may find this approach challenging as they have to cope with tasks that are not clearly defined and work in teams whose members learning needs may not match their own, while academic staff and universities may hesitate to embrace this approach due to its high resource requirements and huge effort needed by the teaching staff (Nepal and Jenkins, 2011).

Reflection in software engineering

It has been suggested that reflective learning goes back to the 1st century and the Roman Emperor Marcus Aurelius Antoninus whose famous work "Meditations" can be seen as a reflection of his role as Emperor, and his life in general (Mac Suibhne, 2009). In modern times, much of the background work in reflective learning was laid by John Dewey in his seminal book "How We Think. A restatement of the relation of reflective thinking to the educative process" (Dewey, 1933). The term 'reflective practice' was coined by Donald Schön in his two renowned books: "The reflective practitioner - how professionals think in action" and "Educating the Reflective Practitioner: Towards a New Design for Teaching and Learning in The Profession" (Schön, 1983, 1987). Schön introduced what he called "reflective practitioner's perspective" and argued that professionals such as architects, musicians and others can benefit from reflecting both on their creative practice and on their thinking about their creative practice.

In the area of software engineering, reflective practice was first introduced by Hazzan (2002) and subsequently was explored by Hazzan and Tomayko (2003), Hazzan and Tomayko (2004), Hazzan and Dubinsky (2009), and others. The main focus of reflection in software engineering is two-fold: (1) improving of "one's understanding about one's own mental processes" in order to improve one's managing of complexities of software development, and (2) increasing "one's awareness of one's own mental processes as well as of the mental processes of others" in order to advance communication among team members in software development teams (Hazzan and Tomayko, 2004).

Problem-based learning in software engineering

Problem-based learning (PBL) was first designed at McMaster University for teaching medical students in the 1960's. Since then, PBL has spread into other universities and other areas, including business and engineering. While there is no consensus among education researchers and practitioners about the precise definition and interpretation of PBL, at least partly because its methodology heavily depends on the area to which it is applied (O'Grady, 2012), in practical terms PBL is commonly understood as a process where students are presented with a problem and independently or through a discussion in small groups they themselves identify what they need to learn to solve the problem. The instructor assumes a role of a mentor while students embark on journey of independent learning to gain knowledge required for solving their problem.

Considering adoption of PBL in computing curricula, O'Grady (2012) argues that the adoption of PBL is superficial and ad hoc, mostly because it is reduced to isolated efforts of teachers striving to improve individual courses. Recent case studies of PBL in software engineering include Richardson and Delaney (2009) and Qiu and Che (2010).

While Problem-based learning is recognised as a powerful strategy providing deep insight and active learning, as well as acquisition of skills for solving real life problems, it is often perceived by students as challenging and could affect their satisfaction with the course. In an interesting study comparing traditional lecture-tutorial engineering courses with their blended project/problem based learning (Pr/PBL) equivalents over two subsequent years, Nepal and Panuwatwanich (2011) discovered that while overall students perform better in P/PBL courses, they rate traditional lectures-tutorial courses more highly, and that at least 1 in 8 surveyed students who believe that Pr/PBL style has improved their job readiness would have preferred to do the traditional lecture-tutorial course instead. This can, in turn, affect the readiness of academic staff to embrace PBL and Pr/PBL, as their career opportunities and promotion outcomes are typically tied with student evaluation of their teaching (Nepal and Panuwatwanich, 2011).

Guided problem-based learning and reflection in SENG3150/3160

The two one semester courses, SENG3150 Software Project I and SENG3160 Software Project II, emerged from a previous one semester third year project course at the University of Newcastle (see Ye, 2009). Students worked in groups of 4 or 5 and used some of the industry software process provided by the our partner Object Consulting Pty Ltd, with whom we have a long standing collaboration (Ye, 2009). At the end of the second semester, representatives of Object Consulting attended the final project presentations and demonstrations, gave generous feedback to students, selected the best project and the runner-up, and presented the winning team with a $1500 prize. This did not have any effect on the marks which were allocated independently by the teaching team.

The project was a real-life software application FlightPub, a website where customers can search for and book airline flights, as well as become members and collect loyalty points. Object Consulting had first-hand experience with this project as they developed similar software a couple of years earlier.

Each week students had 2 hours of lectures, 2 hours of workshops and at least a thirty minute mentor meeting for each team separately. The assessment in our software project courses was in two forms: team assessment, consisting of a few assessment items (e.g., requirements document, testing plan, etc.) and comprising 50% of the final mark, and individual assessment, including PBL report and Reflection Journal.

Throughout each semester, a number of industry seminars were run, some of them presented by University of Newcastle alumni who are now established software professionals, or even run their own successful businesses. These seminars were well received as they motivated and inspired our students, introduced them to the local software industry, and gave them an insight into real life software projects and solution approaches.

Project-based learning naturally lends itself to the software project-based learning environment, as the whole process is focused around the project that is not only a starting point but remains the motivation and purpose of the student learning. In the case of our software project, the separate weekly mentor meeting with each team, in addition to regular two-hour workshops provided an opportunity for regular guidance and feedback from the mentor (tutor, lecturer). Moreover, the first time the course ran we supported students by providing weekly learning goals, as well as recommended resources. However, PBL can be daunting for students used to a traditional lecture-tutorial learning environment. In the rest of this section we will outline the modifications and additions we introduced to the second and third course offerings to make it less confronting and effective for the novice student users.

Guided problem-based learning

One of the main propositions of PBL is that students themselves identify the knowledge they need to acquire in order to solve the PBL problem, which in our case amounts to developing a real life software application. While in their prior years of study students learned the theoretical basics of software development and design, they did not have an opportunity to practice doing it beyond academic lab exercises and assignments. In the ever-changing world of software engineering, there is an overwhelming amount of approaches, practices, issues, development models and potential tools that may be useful for the project. To assist the students in finding their way around these diversities, we introduced an approach that we termed “Guided Problem Based Learning” or GPBL, were we set a PBL “task” to be done and assessed individually but which would nevertheless provide benefit to the whole team and not just the individual student.

More specifically, in Semester 2, 2011 we gave students a set of 3 topics: (1) software and quality metrics; (2) risk management; and (3) software tools. Each student was at liberty to choose a topic, as long as in each team there was at least one student for each of the topics, and preferably two students for software tools. The students were instructed to research their topics and present their findings in a report. Furthermore, the students were encouraged to prepare at least 2 drafts of the report and seek feedback from the lecturer/tutor before submitting the final version of the report. This was used as an opportunity to provide guidance to the students in their independent knowledge acquisition, as well as developing report writing skills. This appeared to work very well for the students, but it was very demanding on the teaching staff, and it was only possible due to the relatively small class size (14).

Using this initial experience, in 2012 the topics were reduced to software tools/technologies only, as these proved to be by far the most beneficial for the project. The teams were encouraged to discuss the tools/technologies and decide which categories are most likely to benefit the project and thus should be explored. The topics should have been chosen in such a way that each such tool category is selected by at least one team member in each team. Each individual report was required to present a brief overview of software tools, clearly show where the selected tool category fits into the big picture and give the pros and cons for using such tools in the project. Then the report should go on to present at least two specific tools/technologies (or, if that is not applicable, at least two alternatives appropriate for the project) and compare them in terms of benefits and drawbacks to the project. The report should then draw clear conclusions as to which alternative should be selected for the project. Again, students were encouraged to prepare at least one draft of the report and seek feedback from the lecturer before submitting the final version of the report.

Reflection

Reflection has been an integral part of SENG3150/3160 from the outset of the course, and it was intensified in each new semester the course ran. In semester 1 2011, 30% of the total mark came from the “Final Report” which was a part of the individual assessment. Students were instructed to describe the context of the project, what they learnt, how they applied that knowledge to the project, the results they achieved, their individual contribution, the limitations of the completed work and how it could be improved/extended.

In Semester 2, 2011, we added an “Individual Reflective Journal” whose name was chosen to emphasise the importance of reflection. Students were urged not only to look back and identify what in their project worked well and what could have been done better/differently, but also what are the things for watch out for, why something came about, and how their understanding/opinion developed over time.

In addition to journals, we introduced diaries for students to fill in and submit each week as a part of their individual assessment. We provided a diary template consisting of tables to record class, team and individual activities related to the course, as well as three text boxes marked “Reflection on the Project”, “Reflection on the Team” and “Reflection on the Course”. The diaries were worth only 4% of the total mark and students would score either 4% if they submitted at least 80% of the diaries, regardless of their content, or 0% if they failed to do that. Therefore, there was little incentive to maintain the diaries in terms of marks, but the students were repeatedly reminded that diaries will be very helpful for the Individual Reflective Journal worth another 10%, and most students were submitting the diaries regularly. Note that the total mark for reflection was reduced from 30% to 14% because we also introduced “Problem Based Learning Report” as a part of individual assessment; this reduction does not indicate in any way a reduction of the emphasis on reflection.

In 2012, the weightings of the assessment items were again revised and the Final Report with a strong emphasis on Reflection, together with diaries and course participation, was allocated 30% of the total mark, and students were encouraged to reflect on the project, team, themselves, and the reflection process itself.

Evaluation

The evaluation of the two courses, SENG3150/3160 Software Project I and II, is based on: (1) Student Feedback on Courses (SFC) collected by the university on all its courses; (2)

student performance in the courses; (3) the extent to which the software tools have been used in the project; (4) the opinions expressed by students in their diaries and reflective journal. We will use this 4-fold evaluation to analyse the progression of student performance, as well as their satisfaction and opinions about the courses, and to evaluate the effectiveness of Guided problem-based learning and reflection in software project courses.

At the University of Newcastle, SFC comprises a number of questions to evaluate the student perception of courses. For each question, students can choose one of the following 5 answers: 1) strongly disagree; 2) disagree; 3) neutral; 4) agree; and 5) strongly agree. In the presentation of the SFC results we will use a number scale from 1 to 5 for the sake of clarity, where typically 5 is the most and 1 is the least desirable. We present student responses over 3 semesters: 2011 semester 1, 2011 semester 2, and 2012 semester 1, where class sizes were 17, 14 and 21 students, and response rates 59%, 43% and 24%, respectively.

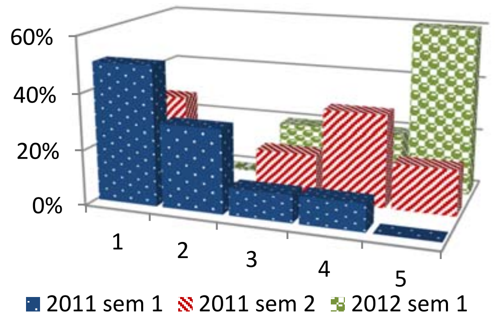

Figure 1: "Overall, I am satisfied with the quality of this course"

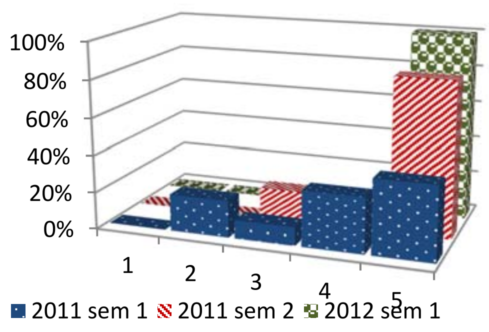

Figure 2: "I made a consistent effort to succeed in this course."

Figure 1 shows the overall student satisfaction with the course and a clear and steady improvement in the student perception of SENG3150/3160 over the 3 semesters.

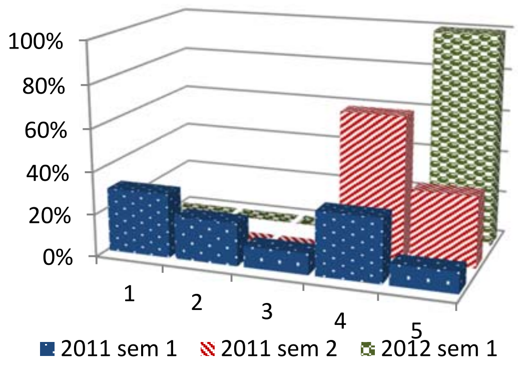

Figure 3: "My knowledge and skills have developed as a result of studying this course."

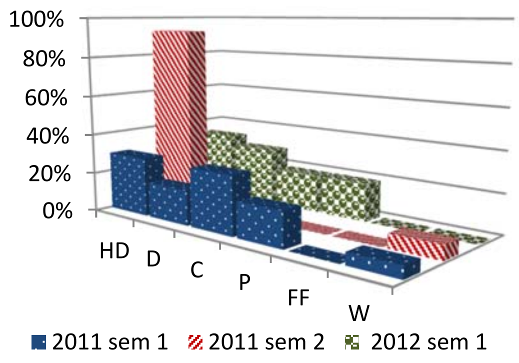

Figure 4: Distribution of students' grades

Figures 2 and 3 illustrate students' perception of their own effort and growth in the software project course. It appears that the students' estimate of their own efforts and acquired knowledge and skills also improved from the first to the third course offering. It also appears that the average rating of the course satisfaction (Figure 1) grows with the average ratio of the effort (Figure 2) and knowledge and skills acquired (Figure 3). It is interesting to compare this with the distribution of student grades over the three semesters (Figure 4). It can be noticed that the grade distribution is very similar in semester 1 2011 and semester 1 2012, while the student satisfaction and perception of their own effort and learning differs significantly. This is consistent with the study performed by Nepal and Panuwatwanich (2011) whose results indicate that there is no relationship between course evaluation and actual performance.

To gain an insight into how much effect guided problem based learning had on student performance we evaluate the extent to which the software tools have been used in student projects. This is only relevant to the semester 2 2011 where the students implemented and tested the FlightPub. Note that using software tools was not compulsory, but it was encouraged and facilitated by the Guided Problem Based Learning task. There were 3 student teams and two of them used at least 4 different software tools/technologies each, while the remaining team used only an Integrated Development Environment (Table 1).

Table 1: A summary of software tools/technologies used by each team

Software tools/technologies	Team 1	Team 2	Team 3
IDE	x	x	x
Version Control	x	x	
Issues Management		x	
Testing		x	
Struts	x		
Hibrenate	x		

Finally, we look at students' reflection journal for more insight into their perception on courses and effectiveness of Guided Problem Based Learning. The prevailing opinion expressed by students was that the Software Project courses were, on one hand, very demanding and time consuming, and, on the other hand, very rewarding and the ones where they learned more than in any other course during their degrees.

I believe SENG3160, be it the most time consuming course I have ever taken, is the truly most important course a graduating Software Developer could take in their degree.

I will remember FlightPub as the project that began my software development career.

There were so many positives to this semester that outweighed the extreme time that was required.

In particular, students felt that the Guided Problem Based Learning where they had to research software tools/technologies was very helpful to their project, and that the tools and new technologies themselves were invaluable, even if their team did not use them. This was generally not the case with the alternative topics of Risk Management and Software Metrics.

Firstly, the introduction of tools (although not compulsory) was excellent.

In reflection, I can see how valuable writing the problem based learning report was. My team actually used knowledge I gained from all three sections of my report in our project.

I would estimate that without the tools (subversion /fogbuz/Junit/eclipse) the group would only have achieved 80% of their potential.

Tools are invaluable. Our group achieved the perfect harmony.

It was about this time of the semester where I began to regret not picking up any extra development tools such as a repository backup, automated testing tools or some other such tool.

Discussion

We looked at the student perception and performance in SENG3150/3160 over the three semesters from Semester 1 2011 to Semester 1 2012.

On one hand, there are no big differences in student performance, as measured by the grade distribution, between two offerings of SENG3150 in semester 1 2011 and semester 1 2012. The grade distribution For SENG3160 offered in semester 2 2011 is quite different from the two offerings of SENG3150, due to the fact that this is the second part of the two semester course where students implement and deliver the project, and typically students are more enthusiastic and committed to the project. Additionally, the members of the weakest team from semester 1 2011 either did not re-enrol in semester 2 2011, or joined strong teams, which affected the grade distribution.

One the other hand, there are big differences in student perception of the SENG3150/3160 courses over the three semesters. Relating the SFC to student opinions expressed in their reflective journals, it appears that the students perceived the SENG3150/3160 as two very demanding and time consuming courses, with huge workload and little guidance, which likely contributed to the early dissatisfaction with the courses. However, the introduction of Guided problem-based learning was perceived as very helpful useful for the project, which is further supported by the large extent to which the software tools have been used in SENG3160 in Semester 2 2011.

There are two additional factors that might have influenced student perception of SENG3150/3160 courses. First, there were altogether there teaching staff, where semester 1 2011 offering was taught by one of the staff, semester 2 2011 by the other two, while the semester 1 2012 was taught by all three staff. Thus the continues improvement of the student perception of the course cannot be explained by potential student preferences for particular teaching staff, but it could indicate that having more teaching staff on board is beneficial for project courses. Secondly, in semester 1 2012 we altered the software methodology to incorporate elements of agile development, which was preferred by students, and thus can account for more favourable evaluation of semester 1 2012 than semester 1 2011 offering, however, it cannot explain the continues improvement of the course evaluation.

Conclusion

Problem-based learning naturally lends itself to software engineering projects, as the initial requirements are somewhat ill defined and all learning is in the service of accomplishing the job. However, pure problem based learning may not be suitable for software engineering projects, as students may find themselves overwhelmed by the complexity of the project and the diversity of possible directions, and thus guided problem based learning appears as a valuable alternative approach.

References

ACS/NZCS/ANZ ICT (2009) Accreditations Manuel. Document 2A: Application Guidelines – Professional Level Courses. Retrieved September 18, 2013, from http://www.iitp.org.nz/news/uploads/PDFs/DegAcc/2-ApplicationGuidelines.pdf

Dewey, J. (1933). *How We Think. A restatement of the relation of reflective thinking to the educative process*, Boston: D. C. Heath.

Graham, R. and Johnson, M. (2012). Australian Computer Society Accreditation Report. Retrieved September 18, 2013, from http://www.acdict.edu.au/documents/RuthAndMike ACSAccreditation ReportJuly2012v3.pdf

Hazzan, O. (2002). The reflective practitioner perspective in software engineering education. *The Journal of Systems and Software,* 63(3), 161-171.

Hazzan, O. and Dubinsky, Y. (2009). Reflection in Software Engineering Education. *Proceedings of ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'09)*, Orlando, Florida, USA , 25-29 October 2009, 691-692.

Hazzan, O. and Tomayko, J. (2003). The reflective practitioner perspective in eXtreme Programming. *Proceedings of the XP Agile Universe 2003*, New Orleans, Louisiana, USA, 51-61.

Hazzan, O. and Tomayko, J. (2004). Reflection Processes in the Teaching and Learning of Human Aspects of Software Engineering. *Proceedings of the 17th Conference on Software Engineering Education and Training (CSEET'04)*, 1-7.

Johns-Boast, L. and Flint, S. (2009). Providing students with 'real-world' experience through university group projects. In *Proceedings of 20th Australasian Association for Engineering Education Conference (AAEE)*, Adelaide, South Australia, 6-9 December 2009, 299-304.

Johns-Boast, L. and Patch, G. (2010). A Win-Win Situation: Benefits of industry-based group projects. In Proceedings of *21st Australasian Association for Engineering Education Conference (AAEE)*, Sydney, New South Wales, 5-8 December 2010, 355-360.

Mac Suibhne, S. (2009). 'Wrestle to be the man philosophy wished to make you': Marcus Aurelius, reflective practitioner. *Reflective Practice*, 10(4), 429-436.

Nepal, K. P. and Jenkins, G. (2011). Blending project-based learning and traditional lecture-tutorial based teaching approaches in engineering design courses. In *Proceedings of 22nd Australasian Association for Engineering Education Conference (AAEE)*, Fremantle, Western Australia, 5-7 December 2011, 338-343.

Nepal, K. P. and Panuwatwanich, K. (2011). Comparative study of project-based learning and traditional lecture-tutorial teaching approaches in undergraduate engineering courses. In *Proceedings of 22nd Australasian Association for Engineering Education Conference (AAEE)*, Fremantle, Western Australia, 5-7 December 2011, 351-356.

O'Grady, M. J. (2012). Practical Problem Based Learning in Computing Education. *ACM Transactions on Computing Education*, 12(3), 10:1-10:16.

Perrenet, J.C., Bouhuijs, P.A.J. and Smits, J.G.M.M. (2000). The suitability of problem-based learning for engineering education: theory and practice, *Teaching in Higher Education*, 5(3), 345-358.

Qiu, M. and Che, L. (2010).A Problem-based Learning Approach to Teaching an Advanced Software Engineering Course. In *Proceedings of Second International Workshop on Education Technology and Computer Science, IEEE*, 252-255.

Reichlmayr, T. J. (2006). Collaborating With Industry – Strategies for an Undergraduate Software Engineering Program. In *Proceedings of SSEE'06*, Shanghai, China, 20 May 2006, 13-16.

Richardson, I. and Delaney, Y. (2009).Problem Based Learning in the Software Engineering Classroom. In *Proceeding of the 22nd Conference on Software Engineering Education and Training, IEEE*, 174-181.

Schön, D. A. (1983). *The reflective practitioner - how professionals think in action.* Basic Books.

Schön, D. A. (1987). *Educating the Reflective Practitioner: Towards a New Design for Teaching and Learning in The Profession*. San Francisco: Jossey-Bass.

Ye, H. (2009). An Academia-Industry Collaborative Teaching and Learning Model for Software Engineering Education. In *Proceedings of the 21st International Conference on Software Engineering and Knowledge Engineering (SEKE),* Boston, USA, 1-3 July 2009.

Copyright statement