Overcoming Challenges in Software Engineering Education: Delivering Non–Technical Knowledge and Skills

Liguo Yu
Indiana University South Bend, USA
Chapter 18
Incorporating a Self-Directed Learning Pedagogy in the Computing Classroom: Problem-Based Learning as a Means to Improving Software Engineering Learning Outcomes

Oisin Cawley
The National College of Ireland, Ireland

Stephan Weibelzahl
The National College of Ireland, Ireland

Ita Richardson
University of Limerick, Ireland

Yvonne Delaney
University of Limerick, Ireland

ABSTRACT

With a focus on addressing the perceived skills gap in Software Engineering (SE) graduates, some educators have looked to employing alternative teaching and learning strategies in the classroom. One such pedagogy is Problem-Based Learning (PBL), an approach the authors have incorporated into the SE curriculum in two separate third-level institutions in Ireland, namely the University of Limerick (UL) and the National College of Ireland (NCI). PBL is an approach to teaching and learning which is quite different to the more typical “lecture” style found in most 3rd level institutions. PBL allows lecturers to meet educational and industry-specific objectives; however, while it has been used widely in Medical and Business schools, its use has not been so widespread with computing educators. PBL is not without its difficulties given that it requires significant changes in the role of the lecturer and the active participation of the students. Here, the authors present the approach taken to implement PBL into their respective...
Incorporating a Self-Directed Learning Pedagogy in the Computing Classroom

INTRODUCTION

Where is the engineering in software engineering (SE)? While there are many technical skills required in the analysis, design, development and implementation of software systems, ask an IT professional to characterize their profession, and you might just as likely solicit the response that they see themselves as an artist, as opposed to a scientist. Given that there is undoubtedly an important design (some might even say creative) element within the practice of SE, it would be reasonable to expect that our SE graduates are also supported in developing non-technical skills.

In addition, if we look at what the academic world has defined under the banner of SE, we clearly see the necessity to arm our graduates with many non-technical skills. Wasserman’s eight notions (Wasserman, 1996), for example, include a software process element, which is fundamental for an effective discipline of SE. This software process element focuses on quality through the organization and discipline within the various SE activities. The Software Engineering Body Of Knowledge (SWEBOK) is currently adding an additional knowledge area, titled “Software Engineering Professional Practice,” which includes “… subareas of professionalism, group dynamics and psychology, and communication skills.” Clearly there is a growing understanding within academia that such “softer” skills play an increasingly important role in the successful outcome of SE projects.

The experiences of two of the authors bears witness to a lot of what has been identified above. OC and IR spent 25 years between them working on SE projects, large and small, in both small and multi-national companies. Their experiences have shown that while technical knowledge is a requirement for much of the SE life cycle, other non-technical skills had been seen to be increasingly important as systems grew in complexity and the business functions became less tolerant with overdue and over-budget projects. Systems complexity, in this sense, is not only a technical concern but also relates to the change in team dynamics as the number of stakeholders and project participants increase. This type of complexity requires oral, written, interpersonal and team working skills that some authors argue our graduates are not being adequately equipped in when compared to their technical abilities (Davies, 2000; Cotton, 1993; Connor and Shaw, 2008). We have recognized that, when using Problem-based learning in our classes, we can provide students with these technical and non-technical skills.

WHAT IS PROBLEM-BASED LEARNING?

“Problem-based learning (PBL) is apprenticeship for real-life problem solving, helping students acquire the knowledge and skills required in the workplace” (Dunlap, 2005). PBL has a long “intellectual history” with its origins in the “philosophies of rationalism and American functionalism” (Dewey, 1929; Schmidt, 1993). Current day PBL emerged in the 1950’s and 1960’s in Case Western Reserve University and McMaster University respectively (Prince & Felder, 2006). In the late sixties, Howard Barrows joined the faculty at McMaster University in Canada. During that time he collaborated with others and developed the approach to learning now called Problem-based Learning (Schmidt & De Volder, 1984). By the early seventies, Problem-based Learning was installed as a total approach to learning and...
Incorporating a Self-Directed Learning Pedagogy in the Computing Classroom

instruction in the Faculty of Health Science at McMaster, with Barrows as its main proponent (Schmidt & De Volder, 1984; Schmidt, 1993b; Barrows, 1986; Barrows & Tamblyn, 1977). Inspired by the success of McMaster, universities around the world introduced Problem-based Learning into their curriculums. These include Maastricht University in the Netherlands, Newcastle University in Australia, the University of New Mexico, Harvard and Sherbrooke University in Canada. This resulted in widespread “cross fertilisation” and networking between the major universities (Barrows, 2000). Problem-based Learning has now spread well beyond the realm of medical education and is now being practiced in other disciplines such as business and engineering (Tan, et al., 2000; Tan, 2003). A number of leading universities now have dedicated PBL Websites. Coupled with this, leading journals on engineering education have dedicated entire issues to PBL (Prince & Felder, 2006).

By the 1980’s and 1990’s, the global economy was changing, increasing the focus on organizational performance, organizational structures and organizational change in general (Hales, 2007; Hallinger, Philip, & Bridges, 2007). Third level institutions started to come under pressure to respond to this level of industrial change and to produce graduates that were capable of operating in this changing environment (Hallinger & Bridges, 2007). A number of universities and practitioners responded to the challenge by implementing Problem-based Learning as a basis of their learning and instruction (Hallinger & Bridges, 2007). Maastricht University established its school of Economics and Business Administration adopting Problem-based Learning as its primary educational philosophy. This was revolutionary, as no examples of Problem-based Learning existed for Economics and Business Administration prior to this (Gijseelaers, 1995). Similarly, Milter and Stinson from Ohio University established an MBA programme in the early 1980’s also using Problem-based Learning (Milter & Stinson, 1995). Early in 1987, Stanford University, School of Educa-

tion implemented their Masters programme for administrators also using Problem-based Learning (Bridges, 1992). In the early 2000’s the University of Colorado introduced a capstone course on software engineering selecting Problem-based Learning as their method of instruction (Dunlap, 2005). Nelson (2003) explores how he successfully taught software development to graduates also using PBL (Nelson, 2003; Prince & Felder, 2006). In the mid 2000’s the University of Limerick (Ireland), and the National College of Ireland implemented Problem-Based Learning at varying levels within their institutions. These implementations ran from full curriculum implementation to single modules across a range of disciplines, including, medical, business, civil engineering and software engineering.

Given the widespread use of Problem-based Learning, it is not surprising that a number of variants have emerged over the years. By the mid 1980’s, the term Problem-based Learning was being used extensively in a wide range of educational methods (Barrows, 1986). Consequently many attempts have been made to explain the concepts of Problem-based Learning (De Graaff, 2003). Barrows (2000) focused on the concepts of student-centered learning, small groups, the teacher as facilitator and the importance of the problem. Barrows alluded to his version of Problem-based Learning as authentic Problem-based Learning (aPBL) (Barrows, 2000; Barrows, & Wee, 2010).

While the Barrows Model (2000) has its origins in the medical profession, it has expanded into many different educational disciplines and has evolved into a distinct educational method (Barrows, 2002; Hmelo-Silver & Barrows, 2006). Barrows (2000) consistently reiterates his core model but was aware of the many variants of Problem-based Learning that had evolved since its introduction into medical education in the mid 1960’s (Barrows, 1996). However he continued to remain faithful to his core model which contained the following characteristics (Barrows, 1998; Barrows & Tamblyn, 1980; Barrows, 2000):
Incorporating a Self-Directed Learning Pedagogy in the Computing Classroom

- Learning is a student centered approach
- Learning happens within the small collaborative group using a structured process
- The teacher operates as a facilitator
- The problem is encountered first and is the main stimulus for learning
- Clinical problem solving skills are developed through interaction with the trigger or problem
- It is through self-directed learning that new information is accumulated

Student-centeredness has its foundation in the theory of social constructivism (Hmelo-Silver & Barrows, 2006). Problem-based Learning facilitates the social construction of knowledge as the learners work through ill-structured real world problems (Schmidt, 1993). Students assume responsibility for their own learning, and work collaboratively in small groups that are not teacher-centered (Barrows, 1998).

Barrows (2000) is very specific regarding the authenticity of the problem. This is also consistent with Dewey (1929) thinking that the problem should reflect real life events and should be the “starting point for learning” (Dewey, 1929; Schmidt, 1993). Barrows stressed the importance of “real patient problems” that the student will face in a work related environment. Barrows (2000) argues that without authentic problems that challenge the students, it will be impossible to develop proper “problem-solving skills.” Hmelo-Silver (2004) agrees with Barrows and alludes to the fact that it has been through cognitive research and experience that Problem-based Learning practitioners have been able to identify the characteristics of good problems (Hmelo-Silver, 2004). However one life-long learning skill that is also developed in a Problem-based Learning environment is Self-Directed Learning (Barrows, 1986; Barrows, 2000).

SELF-DIRECTED LEARNING AND SMALL GROUPS

“Self-Directed Learning is an important approach for Professionalism” (Lahteenmaki & Uhlin, 2011). How could self-directed learning contribute to the development of the Software engineers in terms of their level of professionalism? Barrows (2000) argues that teachers should trust the student to do their own Self-Directed Learning and dig out the material required to problem-solve. Gijselaers and Schmidt (1990) argue that the quality of the problem is of significant importance to the self-directed learning process. They argue that it impacts the amount of time that the student spends on self-study (Gijselaers & Schmidt, 1990). Perrenet et al. (2000) argue that engineering unlike medicine, has a hierarchical structure and care needs to be taken in the case of the self-directed learning process. Students should not be allowed to by-pass any critical topics as incorrect learning of fundamental concepts may impact their understanding of future concepts (Prince & Felder, 2006). Lahteenmaki and Uhlin (2011) argue that reflection plays a large part in the self-directed learning process. They explore the principles of cognitive psychology argued by Gijselaers (1996) to explain that learning is a construction from prior knowledge and that reflection plays a large part in the learning process (Lahteenmaki & Uhlin, 2011). While students spend time on self-study, they also work collaboratively in small groups. Barrows model (2000) suggested a group size of five to eight - or even nine - students (Barrows, 1996). However, Gijselaers (1996) uncovered situations where the group size was increased to twelve (Gijselaers, 1996). Barrows (1996) accepted a large group size, but only under particular circumstances and in a very controlled environment (Barrows et al., 1986). A new phenomenon has arisen in Problem-based Learning which may be
Incorporating a Self-Directed Learning Pedagogy in the Computing Classroom

called “Small Group Creep”: adding one more group member because it will not make a differ-
ence (Gijselaers, 2011). The concern here is that the benefits attributed to Problem-based Learning
and small group learning will be lost in the inter-
est of institutional economics and cost saving.
This could affect other Problem-based Learning
resources such as those of the facilitator.

PROBLEM-BASED LEARNING IN SOFTWARE ENGINEERING

Software Engineering can be seen as a technical
subject in which students are expected to develop
skills such as programming, software and systems
design, architecture design and networks. How-
ever, these skills are no longer sufficient for a world
of work which requires software engineers to col-
laborate with others, to understand problems and
to work in cross-functional domains with which
they would not be familiar. Richardson & Hynes
(2008) argue that curriculum developers need to
provide both content and processes that develop
specific sector skills. In so doing, institutions
would go a long way to preparing students for
the commercial environment that they are facing
into (Richardson & Hynes, 2008).

Therefore, the education of Software Engineers
for the 21st Century requires more innovative
approaches then the traditional didactic method
doing at (Vat, 2006). Traditional Software
Engineering courses are often accused of stifling
students’ independence and imagination (Vat,
2006). In some cases, tutors have selected projects
and team leaders but by in large have ignored the
application of real world problems (Shim et al.,
2009). Today’s Software Engineering graduates
require a wide range of characteristics including,
teamwork, ability to work under pressure, customer
focus and the desire for continuous learning and
self-oriented learning (Shim et al., 2009; Vaughn,
2001). Therefore, it is easy to understand why
software engineering students feel that software
ingineering is complex, requiring as it does social
skills as well as technical competencies (Shim et,
al 2009). One pedagogical approach which can
address the challenges facing software engineers
is Problem-based Learning (Dunlap, 2005; Vat,
2006; Shim et al., 2009). While Vat (2006) argues
that software engineering education has always
used well-defined problems, a change in mindset
is required. He suggests a need for collabora-
tion, skills development and lifelong learning as
opposed to the fixed stop-start nature of current
educational practices.

At Colorado University the developers of the
capstone course selected PBL as a method of
instruction as they considered that there was a
strong line between the Software Development
Life Cycle (SDLC) and PBL (Dunlap, 2005). Their
aim was to expose the students to the real world
of software engineering. This involved interaction
with a real client, the formation of a software
engineering project team and the preparation of
the request for a proposal (RFP). In their course
design Dunlap and her team matched the stages
of the Barrows model to the SDLC model as they
considered both models reflected the type of activi-
ties Software Engineers would be exposed to in a
real life project (Dunlap, 2005). Richardson and
Delaney have also reported on their use of PBL
for educating MSc students in software process
quality (Richardson & Delaney, 2009, Richardson
& Delaney, 2010).

Introducing problem-based learning into the
software engineering classroom takes time and
commitment not only from the tutors’ and students’
point of view, but also from the institutions as a
whole. Preparing software engineers for the 21st
century may not be easy but the positives will out-
weight the negatives. This could be achieved by
using innovative and inductive teaching methods
such as PBL.
THE PBL IMPLEMENTATIONS

Introduction

In this section we describe in detail two case study PBL implementations. Although both implementations advocate the same learning and teaching pedagogy and have comparable class sizes, it is important to point out that they are performed in different organizations, with differing student profiles and assessment strategies. The NCI case study was focused at an introductory class (2nd and 3rd year computing) while the UL case was more advanced (MSc and 4th year computing). PBL assessment within NCI case was confined to 40% of the module marks, while in the UL case it was 100%. Interestingly both cases had a good mix of international students and also students with some work experience, with the MSc course in UL being the most culturally diverse. These differences are worth bearing in mind, since they affect the way in which both lecturer and students interpret and engage in the learning process. Dahlgran and Dahlgran (2002) argue that the learning outcomes have a significant influence on the students study strategies. Through their empirical research on three academic programmes at Linköping’s University in Sweden, they have shown that not all academic programmes use learning outcomes in the same way. The variation of how the learning outcomes were used by the students and their intended use by faculty unearthed a potential difference in “educational culture” and student’s interpretation of problem-based learning. This may be an aspect that could be explored in our future research.

Department of Computer Science and Information Systems, University of Limerick, Ireland

Overview

PBL has been used by one of the authors (IR) as the method of teaching Software Quality and Software Process Improvement to MSc in Software Engineering and 4th year BSc in Computer Systems classes since academic year 2009/2010. Having initially introduced PBL to a second-year undergraduate class (Richardson & Delaney, 2009), she recognized its potential as a teaching method for more senior classes within the department.

The Software Quality and Software Process Improvement modules were initially taught to MSc and 4th year students for 2 hours over 12 weeks with supplementary 1 hour tutorials as required. Lectures were generally presented on PowerPoint slides. Although discussion was encouraged, the lecturer did most of the talking. Inter-student interaction was minimal. Journal and conference research papers were assigned as reading material, but were rarely read by students. Up to two lectures were presented by guest lecturers, generally from a software engineering project manager. Within this environment, classes were seen as theoretical, uninteresting for lecturer and students, and students found it difficult to understand software quality and process concepts. The lecturer did not observe that students understood the topic nor its importance, and was concerned that they completed the module without an in-depth understanding of what is really meant by ‘software quality’! Assessment for the module was divided between a team project (40%) and final exam (60%). Project teams were self-selected, worked outside of class time, and presented a final paper at the end of semester. During the semester, the project was never discussed in class, and any learning was not shared within the class. There was no record of individual involvement in the project, nor was individual’s participation identified. The project was normally a case study requiring domain knowledge which the students were unlikely to have, such as manufacturing or finance. The final exam dealt with concepts presented in class. While those students who did the assigned reading performed well in the exams, there was no incentive for students to actively research for the module. No advantage was taken of student background and experience.
Software Quality and Software Process Improvement PBL Modules

In the Department of Computer Science and Information Systems at the University of Limerick, PBL for Software Quality and Software Process Improvement (SQ/SPI) modules has been implemented five times since academic year 2009/2010, twice with 4th year classes and three times with MSc classes. Class sizes have ranged from 14 to 28 students who come from a variety of backgrounds - full-time/part-time students, many years/little or no industry experience, Irish/international students, prior/no prior PBL experience and native/non-native English speakers.

As previously stated, the success of the PBL curriculum is dependent on the development of a good problem. Potential problems were considered, focusing on the requirements for an engaging and interesting problem which would motivate the students to look for a clear and deep understanding of SQ/SPI concepts. It should also relate to a familiar situation allowing students to focus on solving the problem rather than on understanding the domain. For these reasons, e-Health software quality research was identified. As IR (lecturer) was researching e-Health, use of this topic would be beneficial to her facilitation of the module, also having the advantage of bringing her research to the students. The problem trigger was presented to the students during the second week of the module. It involved the students viewing an online video titled “Emergency Department – A Day in the Life”\(^2\). The students were then asked to develop and write the software quality plan for a hospital.

As the video commences, a patient is taken in from ambulance on a trolley into a hospital. This is the last time we see any patient. The focus is on hospital computer hardware systems, such as bedside monitors, and on staff discussions around computer screens. Just watching this video a few times in class allowed for discussion around the use of computing equipment and medical devices in hospitals, the realization that where there is hardware software is also present, and further discussion on software quality required by safety-critical healthcare systems.

PBL’s introduction led to changes in class organization. A two-hour lecture was used. Students were split into groups of four, with three or five students in some groups depending on numbers. International students were considered. Depending on class make-up, groups in some classes consisted of students from one country, while in others, there was a requirement that groups would be global, with a mix of nationalities and language in the group. During each scheduled session, students joined their groups immediately to work on the problem. The lecturer’s role changed to that of a facilitator. She circulated between the groups, discussing issues that arose, ensuring that all groups worked towards a relevant software quality plan by directing them towards relevant research. On occasion, she gave 10-15 minute lectures on specific topics. For example, one lecture ensured that students understood the characterization of processes as Organization, Management, Engineering, Customer-Supplier and Maintenance processes, thus removing the exclusive focus on Engineering processes. Additionally, at the end of class, group discussions were summarized during a short 5-10 minute discussion with all students.

During group discussions, as in PBL theory, students filled specific roles: discussion leader, recorder, observer and team member. They kept minutes of meetings and reviewed these each week. Actions from the previous week were discussed and students circulated papers that they read since the previous session. They had Internet access and freely viewed papers or other information they needed during the meetings. Some groups stayed in the classroom to conduct meetings; others moved to the adjacent café to hold their meeting and discussion. At the end of each meeting, actions for the following week were distributed. Discussions within the groups were varied and interesting. Students discussed personal situations where they had seen software and hardware system
Incorporating a Self-Directed Learning Pedagogy in the Computing Classroom

use in hospitals. These included observations at the lack of concern for privacy of patient data, the lack of integration of patient data, and the copying of data from medical devices to paper charts. In the literature, while they found that regulation is integral to the production of medical device software, they noted that regulations are not observed within many healthcare situations. In addition to these discussions, to ensure an understanding of quality requirements, and to give students an insight into the hospital quality system, a clinical quality auditor from the Health Service Executive visited the class after they had researched the problem for 4 weeks. She gave a short presentation followed by a 90 minute question and answer session with the students. She was able to give them further examples regarding how software is used within hospitals and what development practices are used there.

The problem-based learning modules have been continually assessed with no final exam. For assessment, a group paper (25%) and two presentations to the full class (12.5% content, 12.5% individual presentation skills) are required to demonstrate the students’ knowledge of the concept of software quality. This knowledge includes the ability to discuss regulations and software processes. Presentations are reviewed by IR and another lecturer who is familiar with the topic. For class participation (10%), IR observes whether students are bringing knowledge from external sources and how well they engage with other group members. Students are also orally examined individually four times, each worth 2.5%. An example would be to have individual students present the group’s progress to date. The final part of the assessment is a presentation of an individual portfolio (30%). This includes summaries of papers read, a personal reflective journal, meeting minutes, and an outline of individual project participation.

A summary of the student and lecturer experience is described in the next two sections. This summary was collected from discussion within the classes, formal interviews with some students, reflective journals kept by students and lecturer, and informal feedback from individual students.

Improvements through PBL

SQ/SPI classes are now very interactive with input from students and lecturer alike. Previous industrial experience, medical experiences and international experiences are brought into the discussion and learning by the students, one of whom has described PBL as a very interesting and innovative way to learn. Additionally, students regularly receive feedback from their peers, from the lecturer as facilitator and from their assessments. It has been very important to students to have the subject matter expert (clinical quality auditor) available for the question and answer session. This gives them an opportunity to meet someone who is working at the coalface, who is very knowledgeable regarding the importance of good software quality in healthcare and presents an understanding of the difficulties that arise when quality is sub-standard. This session, held at a pivotal point in the module, has been recognized by students and lecturer alike as being invaluable to the groups’ progress.

Both students and lecturer are enjoying the classes, and they have given the students …an opportunity to get to know the rest of my classmates better... They have a real sense of solving a problem, and they are learning from each other in a “Student” way, while also putting in more work... Through reading and reviewing academic papers, discussion, peer learning, facilitation and the short lectures given, student knowledge has increased. This is obvious through assessments and reflective journals, and was not observed when this module had been taught previously. Students themselves recognize this: Personally, I believe that I have learnt more through PBL in the first 8 weeks than I would have in a standard classroom environment. They also notice that...the things you learn through …stay with you longer.
Incorporating a Self-Directed Learning Pedagogy in the Computing Classroom

Student attendance has improved, and students are very conscious of disrupting their group if they are unable to attend for a particular reason. Students work consistently, and each week it is noticeable that the groups are progressing with their projects. Students have been reviewing academic papers, which is a requirement for this level, but something which has not obviously been undertaken in the past. In addition to learning about SQ/SPI, students have the opportunity to acquire soft skills, which have become very important for software engineering students. We presented evidence previously (Richardson et al., 2011) that students’ skills in communication, teamwork, problem-solving, decision-making, leadership, management and time management have improved through participation in the PBL SQ/SPI module. In addition they brought a work ethic and motivation to the module that was not seen previously.

Difficulties Implementing PBL

Although this case describes where PBL was introduced with groups of senior students (MSc and 4th year), they had not normally attended any PBL module previously. Therefore, it can be difficult to get students into the process at first. This was particularly true when the class was mainly international and not native English speakers. In some cases, their prior education seemed to be very much at odds with what was required here, for example, self-directed learning, and students found this concept difficult to grasp. This required intensive work as facilitator to get the project started within the class. All students who participate have to understand their role within the group, the roles were rotated from week to week. However, this caused problems and maybe if they retained the role for a longer time period there could be some continuity and people could get immersed in the given role. There is also recognition that their active participation in the problem was the key to their learning and when people did not become involved sharing of the knowledge is reduced. This is also true of group projects, and in the PBL situation due to the interaction in class and regular feedback can be more controlled than in the traditional classroom. However, once students realized that lack of participation caused significant problems and was being actively monitored by the lecturer, their work rate improved and consequently their progress in the module improved.

Additionally, there was a requirement to carry out assessments throughout the semester. This consisted mainly of oral examination and observation of the students in their work. As this was not the normal way of assessment, this proved quite difficult for the lecturer.

Another concern was whether this concept suited all those involved in the class. We recognize that the same learning technique many not be universally successful, and this was also noted by the students: I don’t think it suits some people in my group.

PBL within SQ/SPI

Using PBL within the SQ/SPI module should allow for the:

- Provision of an understanding of software quality and software process improvement concepts;
- Provision of soft skills such as teamwork, communication and problem solving;
- Introduction to up-to-date research, demonstrating how this could potentially be useful to students’ in the future.

Taking each of these points into account, the implementation of PBL into the module has been successful. It is not without its difficulties, and within the context of class profile, the mode of implementation sometimes has to be modified as the module progresses. However, when one considers this compared to the traditional lectur-
Incorporating a Self-Directed Learning Pedagogy in the Computing Classroom

Incorporating a Self-Directed Learning Pedagogy in the Computing Classroom

ing mode, lecturers can see that PBL shows up the difficulties experienced much earlier in the module, and changes can be made before the final examination, which is often where lecturers realize that students have not attained the knowledge they strive to impart.

School of Computing, the National College of Ireland, Dublin, Ireland

Overview

In the School of Computing at National College of Ireland we were faced with the same problems that many Higher Education institutions seem to struggle with. While students did well in exams and continuous assessment, employers of graduates felt that some students lacked communication and problem solving skills that are essential for the job roles they were offering. We were looking for a structural change in our teaching that would help to develop these skills further in our students. Facilitated by a visiting researcher who is an expert in PBL, we conducted some preliminary trials in 2009. Starting from the academic year 2009/2010, we converted several modules to PBL including subjects like Programming, Software Engineering, Artificial Intelligence and Personal & Professional Development. We were hoping to enhance students’ skills development, but also to increase their motivation by applying new concepts to real life problems.

Today, we are delivering a range of modules through PBL to about 300 students each semester. In this section we summarize our experience with the implementation of PBL and reflect on the issues that may arise. We begin with an experience report which describes a typical implementation of PBL for Software Engineering. We provide data on how students experience the PBL process and how their assessment results are affected. To conclude we discuss the perceived strengths and weaknesses of using PBL for software engineering education and illustrate the barriers encountered so far.

Exemplary PBL Implementation Experience

This section summarizes one author’s (OC) experiences with the implementation of PBL in a Software Engineering module. The module ‘Introduction to Software Engineering’ was taught to a combination class of second year students on the BA in Management of Technology in Business (BAMTB), and Higher Certificate in Computing (HCC) courses. In total there were 48 students (BAMTB 21, HCC 27). The module had three contact hours per week, which would typically be allocated as a two hour lecture and a one hour tutorial. To incorporate a PBL approach, the assessment strategy included a project component. The project was worth 40% of the final grade and, using a self-directed learning approach, required the class to work in groups and submit group projects.

Most students would already have had some introduction to the concepts of PBL from their first year, however, during the introductory session it was clear that some students did not know what the learning approach entailed, and other students seemed interested in (re)hearing the historical and theoretical background to the pedagogy.

Following this introductory session, the imperative was to form the class into groups. This was found to be a somewhat difficult task. A very important consideration is the size of the groups, with literature suggesting group sizes of 4 or 5 being effective (Delaney & Mitchell G., 2002). The lecturer allowed slightly larger groups to form not fully realizing the possible consequences this can have. The average group size came to 5.8 members. He also allowed the class to form their own groups, and as is to be expected some students were not able to find a group to join. In the end OC had to form a new group composed of just 3 people.

Central to a PBL approach, a trigger was introduced for the project which was the YouTube video of the cinematic trailer for the computer game ‘Assassins Creed: Revelations’®. The stu-
Incorporating a Self-Directed Learning Pedagogy in the Computing Classroom

Students were then told that they had to design and build (to a prototype stage) a computer game for their project.

The formative assessment strategy consisted of specific software engineering artifact delivery every two weeks. The schedule was as follows:

- **Week 3:** Group Submission of a Requirements Specification (20%)
- **Week 5:** Group Submission of the Analysis Diagrams (20%)
- **Week 7:** Group Submission of the Architecture and Design (20%)
- **Week 9:** Group Submission of Prototype implementation (15%)
- **Week 11:** Group Submission of Test Plan, Unit Testing (20%)
- **Week 13:** Group Submission of a presentation and demonstration of a working prototype (5%)

Each two-week period began with a specific trigger indicating the deliverable for that section of the project. For example, the first deliverable was for requirements specification, and the trigger was a ‘Dilbert’ type cartoon depicting a manager telling a developer they do not have time to gather product requirements so they should just start developing the system. Similar triggers were used for the other phases, but care was always taken to ensure the content of the trigger was both relevant and instructive for the students.

The trigger session was followed by a one week period allocated for students to work within their groups towards an understanding, and development, of the particular deliverable. OC, as facilitator, monitored the groups’ progress, discussed any specific questions the groups had, and held short impromptu clarification/instructive sessions if a particular issue identified was relevant to the whole class.

During the second week, the lecturer delivered what was referred to as a ‘landscape lecture’ on the relevant topic. This would include any theoretical and practical components of the subject. Tutorial sessions were also scheduled where the students were required to generate the necessary artifact for a sample project, thereby assisting them in their PBL projects.

As part of a formative assessment strategy, each of the deliverables was reviewed by the lecturer and feedback given to each group in the following week. It is important to ensure feedback is given as scheduled and is constructive in nature. With the time demands of a typical lecturer at NCI being quite high (due to teaching multiple courses), it can at times be difficult to adhere to that process, but if the lecturer misses or delays feedback it can disrupt the learning process since the students start to move onto the next deliverable.

One other important aspect to the PBL project was that each team was asked to keep a journal of the group’s activities. They were asked to record the important group activities such as team meetings and who attended, questions/topics that they felt they needed to research or ask the lecturer about, the tasks assigned to each team member, and any team issues that might have arisen. The journal was to be updated weekly and would be used at the end of the course to assist in the marking process. To facilitate this we made use of Moodle’s online group folder functionality which allowed the lecturer to set up individual group access to a private area on Moodle. Groups were only able to see their own journal entries and each member was able to add their own comments to their group’s journal.

**Student Experience**

Although most students engaged fully in the process initially, there were some negative attitudes which quickly began to surface. The realization that they really would not be getting direct answers to their questions was something that they were unaccustomed to and dissatisfied with. All
Incorporating a Self-Directed Learning Pedagogy in the Computing Classroom

student questions were listened to and guidance was given by the lecturer, however, feedback from the students clearly indicated that they felt they needed more direction. The skill required of the facilitator is to be able to balance this student desire to be told the answer, with the PBL methodology which calls for guidance, discussion, and explorative study by the students.

The students worked in groups and would assign tasks to each other for research or development, and bring the results of that back to the next group meeting and/or update the group’s journal on Moodle. Learning was evident through this but as expected, some groups worked better than others and at times group members felt they needed to consult with the lecturer about the lack of engagement from other team members. An aspect of Software Engineering is being able to work within groups and deal with these types of issues, so as facilitators we need to encourage the groups to resolve these types of issues internally within the group. Interestingly, some students reported that they did not necessarily want to get any team member in trouble, but the fact that a lack of engagement from other members could affect all their project marks, was something they were not prepared to tolerate. This aspect of the project, individual versus group marking, is something we will return to in the next section.

The final deliverable for the group was the full set of updated SE artifacts and a group presentation of their project and demonstration of their prototype. This was both challenging and enjoyable for the students. The challenge came in pulling all the individual contributions together into a cohesive package. As with many industry SE projects, multiple team members will have been assigned individual tasks which will need to be integrated into the final product, so this gives the students some practice in this area. The enjoyable part came in the form of the group working together on developing and presenting the prototype they had designed. A sense of pride was clearly evident as the team members became inventive and resourceful in developing and choreographing the presentation. Again, this was an excellent exercise in a typical SE prototype demonstration to stakeholders. This, however, was one occasion where the group size became an issue. With larger groups it was difficult to ensure that everyone contributed to the presentation, and it therefore becomes more difficult for the lecturer to assign individual marks.

Assessment

Assessing a PBL module, in this case the project, can be a difficult task. The nature of the PBL learning process is inherently about learning and working within a group context. The Irish Third Level examination process, however, is about individual marks and we therefore have to find some way of allocating individual marks to each student. The lecturer’s approach was to use a combination, thereby rewarding a good group effort but also rewarding the individual contributions which show evidence of achieving the learning outcomes of the module. This was achieved by utilizing a detailed grading rubric which broke the deliverables down into specific components with allotted marks. For example Figure 1 depicts the requirements engineering section of the rubric showing the breakdown of the marking scheme for that deliverable.

Having learned from assessment difficulties in previous years, at the start of the project the class was instructed to break up their proposed project into functional elements and assign one to each team member. They were each to deliver all the requisite artifacts for their own part of the project but present them all together into a cohesive final deliverable. This way the lecturer was able to allocate individual members marks based on what the group submitted. The students had access to the marking rubric from the start, and were therefore in no doubt about how the project...
would be assessed. The group presented their project in a stand-up presentation and the lecturer posed questions to individual group members to ascertain their involvement and depth of knowledge. If it was evident that the group worked cohesively together and each member demonstrated competence in the various aspects, then a single group mark was awarded and each member received the same mark. However, if this was not the case then the project deliverables were examined in more detail to ascertain what mark each member should be awarded.

Lecturer Experience

PBL requires a mindset change in the lecturer. The first thing to understand is that the lecturer’s role shifts towards that of a facilitator of a student-centered learning process. In fact, the process should not only be student-centered but student-driven. Accordingly, the lecturer needs to encourage students to seek the knowledge they require by getting them to pose questions, discuss different aspects of the topic within their groups, and assign roles and tasks to each group member for individual research. This is a very different role to the common didactic (lecture) style of teaching, and it requires perseverance. There is a great temptation to give the answers to student questions as this brings immediate satisfaction to the student and lecturer. However, this bypasses the learning process inherent in a PBL process.

An additional activity which the lecturer performed, similar to the groups’ journals, was to keep his own journal of events or observations which he as the facilitator experienced. Since this also requires some personal reflection, some effort is needed to remember to keep adding to the journal. It proved to be beneficial in enhancing the academic review process at the end of the module.

Progression

In the following Semester, students encounter a follow-up module to “Introduction to Software Engineering” called “Object-Oriented Software Engineering.” As students are already familiar with the PBL process at that stage, the induction session can be reduced. The module is focused on the Unified Modeling Language (UML). Accordingly, groups are required to produce sets of UML diagrams for a given system. They receive a series of landscape lectures as input, providing an overview of a particular method, but students then have to explore the details on their own initiative and find out how each method applies to their specific project. In line with the requirements of the previous module, students have to document the learning process and reflect on their learning in an on-line forum. A total of 40% of the total mark was assigned to the PBL project, the remaining 60% were assessed through an exam.
Student Feedback and Performance

We collected feedback from students on their learning experience as well as assessment results for the “Object-Oriented Software Engineering” module. Of the 51 students in the 2011 cohort, 25 responded to an on-line questionnaire sent out at the end of the Semester. The questionnaire comprised a set of rating questions on their experience and progress on a five-point scale (1: “not at all” to 5: “very much”). We also asked them open-endedly to list what they liked about the module and how it could be improved.

Overall, the feedback was not as positive as we would have expected initially. On the one hand, they indicated that they felt the approach promoted teamwork and that they participated actively in discussions (x=3.91). On the other hand, they rated the improvement of their critical thinking, problem solving and communication skills as neutral. They frequently referred to the change in lecturer behavior. When asked how to improve the module, students suggested that “guidance from the tutor could be improved” and “lecturers to assist more in solving problems.” Moreover, when comparing their experience to a traditional course, students felt that they had learnt the subject less thoroughly (see Table 1).

This is however in contrast to the actual results. As can be seen from Figure 2, the actual assessment results based on similar exams and similar continuous assessments remained stable in comparison to the baseline of 2009. A one-way ANOVA reveals that the observed differences in the overall results and in exam results are within random variation. The only statistically significant change was in fact an increase of the continuous assessment results between 2009 against 2011 (p<0.012) and 2012 (p<0.023). This means, that students feel that they learn less, but their actual results remain largely the same.

Table 1. Results of feedback questionnaire for the module Object-Oriented Software Engineering on a five-point scale (1: “not at all,” 5: “very much”)
DISCUSSION

The Right Attitude

Barrows (1998) argues that student-centered learning can be “destroyed if not weakened” (pp 630-633) if it is bolted onto an otherwise traditional based curriculum. Within UL and NCI this is what was done with other computing modules, and indeed some of the NCI case study module, being delivered through a typical lecture style approach. As described, some negativity concerning the PBL process was experienced, however, this is not usual with the introduction of PBL, “…in practice, the self-directed learning of students is sometimes confined by the teacher’s limited understanding of the learning styles, past learning experiences and aspirations of the students.” (Chung & Chow, 2004). We feel it is important to consider the students’ background when examining this. When we view it from within the context of the Irish Primary and Secondary education system, where the predominant pedagogy tends towards authoritarian didacticism, then we may understand where this student frustration originates. Within Ireland, students at third level typically will not have encountered PBL before, whereas in some other countries they have made strides to incorporate it into the curriculum for secondary and even primary schools (Belland, 2010), (Kolodner et al., 2003). Holland, for example, is home to Maastricht University (MU), one of the first Universities to teach solely via the PBL method. Students who apply to MU are in no doubt about the teaching approach they will encounter. We suggest, therefore, that it would be interesting to compare the attitude of third level students to the PBL process between Ireland and Holland for example. This would be informative in terms of understanding how prior knowledge and exposure to the PBL process can help or hinder its use within the third level setting.
The ‘Problem’ with PBL

Both case studies clearly reported that students were looking for more direction. We feel this is partly a consequence of the previous point. While it is difficult for a lecturer (now facilitator) to do this, the provision of information other than that which is critical to get students starting work on the problem should be avoided. With a well-structured problem they should be able to reach their learning outcomes independently. The importance, therefore, of the problem trigger is something which needs to be highlighted. “Well designed and authentic problems are crucial to the success of PBL” and should be “…authentic, engaging, deliberately loosely-structured, linked to learning outcomes and key concepts, multidimensional, and graduate attributes and professional practice focused” (O’Grady et al., 2013). This requires some careful consideration within a SE context, where practitioners are not equipped with an established list of worked examples as they are in the Medical profession. Nonetheless, the literature for PBL in the SE domain is expanding and a growing body of knowledge on problem development and validation techniques is developing.

Breaking the Rules

Although PBL advocates self-directed learning, it is interesting to note that both case studies here incorporated short and specific lectures as part of the process. At certain points it was thought necessary to delve into a particular point to either clarify something, share the knowledge with the whole class, and/or direct the class in some manner. However as tutors that have embraced an inductive teaching pedagogy we would argue that students have different learning styles (Kolb & Fry, 1975; Prince & Felder, 2006). We used different interventions within our PBL cycles to scaffold and support the students in their learning process. Another proponent of student scaffolding is Lev Vygotsky. Within his social constructivist view of development he argues that through collaboration and dealing with real problems true learning takes place (Harland, 2003). Vygotsky also argues in favor of the zone of proximal development, explaining that if you expose students to learning without the proper scaffolding i.e. outside their development zone then you will lose them altogether (Harland, 2003; Prince & Felder, 2006). Therefore one form of scaffolding in our PBL cycle was the use of short lectures.

Importance of Assessment

What is the best way to assess within a PBL environment? This is an interesting question given that PBL is learning within a group environment, but we as educators must provide individual assessments. Do we assess at the group level where each team member receives an equal grade? Such an approach evokes calls for fairness from students who feel they get penalized for the poor work of others. Indeed in an ever more competitive workplace for graduates, hard working students query how they can outperform their classmates if their individual effort is not being fully rewarded. Or should we assess solely at the individual level? In this case there is an argument for students to concentrate on individual learning, counter to the team working skills we would also like them to develop.

In defining the process of assessment Huba and Freed (2000, p. 8) explain that it is “the process of gathering and discussing information from multiple and diverse sources in order to develop a deep understanding of what students know, understand, and can do with their knowledge as a result of their educational experiences; the process culminates when assessment results are used to improve subsequent learning” (Huba & Freed, 2000; Levia & Quiring, 2008). In both the traditional teacher centered and Problem-based Learning approach, assessments fall into two main categories namely formative and summative (Levia & Quiring, 2008). However assessment
within a student centered pedagogy such as PBL needs to be carried out in a different manner than those of a traditional teacher centered environment (Ramsden, 1992; Knight, 1995; Levia & Quiring 2008). Within a PBL cycle students establish their own learning outcomes, therefore, regular assessment within this process is required to ensure that the students achieve their course objectives (Mitchell & Delaney, 2004; Levia & Quiring, 2008). In addition, because students are working on authentic problems representing real world issues, an authentic assessment strategy concentrating on the development of critical thinking and higher order skills development is required (Tai & Yuen, 2007). There are many variations of an authentic assessment strategy: these include, assessing performance within the tutorials process, the generation of a portfolio analysis and the preparation of a reflective learning journal and finally peer and self-assessment (Hart, 1994; Phillips, 2005; Tai & Yuen, 2007). Assessments of this nature require students to engage in collaborative practices with strong team and communications skills in order to reach a resolution to a complex problem (Tai & Yuen, 2007).

The cases presented in this chapter have each included both group and individual assessments. In addition, both modules were taught as part of overall courses in which there are marks given for both individual and group work. This helps to ensure a balance within the modules and the courses.

Extending the Programme

Rolling out PBL on a larger scale, for example across a School, Department or Faculty, is a different prospect than a single module pilot implementation by enthusiastic lecturers. When proposed in NCI it was noticed that some faculty members were hesitant to adopt the new approach. Two support workshops were organized to introduce faculty to PBL and to help them convert their modules. Nevertheless, some lecturers felt that either their subject area was not suitable for delivery through PBL or that negative reaction of students in other modules had put them off. It is important that all possible support is given to faculty new to the PBL approach.

Within NCI, to assist faculty members in getting started with PBL, two new support mechanisms were developed. First, a PBL induction session was designed that would familiarize students with the PBL process, establish ground rules in the groups and assign the roles. It consisted of a set of problem solving and communication exercises where students could practice their skills and become aware of the difficulties that can arise in group work (Weibelzahl & Lahart, 2011a). Secondly, a “PBL toolbox” was developed: Each group receives a deck of 30 cards. Each card refers to a key concept or group activity that has been introduced in the induction session. Group members and facilitators can “play” these cards during discussion to bring the group back on track or to facilitate better learning (Weibelzahl & Lahart, 2011b). Lastly, a Web-based resource was created that makes all the exercises available online and searchable (see Figure 3). Lecturers can select the skills they want to address in their induction session and then choose from the available exercises. Lecturers can also rate and comment on resources. New resources can be added through an on-line interface. Currently, there are about 100 exercises available.

Similarly, the continued implementation and development of Problem-based Learning at UL was formalized and strengthened with the development of a Community of Practice (CoP) in 2011. Through the CoP a series of staged workshops were run to train faculty and tutorial staff in the concepts of PBL. In order for these to be fully effective, the CoP identifies and invites workshop facilitators with first-class national and international expertise in the area of PBL in general, and problem (trigger) design in particular. The PBLCoP is in the final stage of developing a CoP Website. This Web site will allow the dis-
semination of PBL news through the University of Limerick and the wider community. PBL-related journal articles have been gathered together and will be presented on the CoP Website in a single repository; this is in addition to plans for future workshops. Funding for the creation of the PBLCoP and Web site were made available through a quality in teaching initiative (QIFAC) at the University.

ACKNOWLEDGMENT

This research is supported by the SFI Principal Investigator Programme, grant number 08/IN.1/I2030 (the funding of this project was awarded by Science Foundation Ireland under a co-funding initiative by the Irish Government and European Regional Development Fund), and supported in part by Lero - the Irish Software Engineering Research Centre (http://www.lero.ie) grant 10/CE/11855.

REFERENCES


Incorporating a Self-Directed Learning Pedagogy in the Computing Classroom

Barrows, H. S. (2002). Is it truly possible to have such a thing as DPBL? *Distance Education*, 23(1). doi:10.1080/01587910220124026


Incorporating a Self-Directed Learning Pedagogy in the Computing Classroom


Incorporating a Self-Directed Learning Pedagogy in the Computing Classroom


ADDITIONAL READING


Engel, C. (2009). An Internet guide to key variables for a coherent educational system based on principles of Problem-Based Learning. *Teaching and Learning in Medicine, 21*(1), 59–63. doi:10.1080/10401330802384888 PMID:19130388


**KEY TERMS AND DEFINITIONS**

**Formative Assessment:** The monitoring of student learning to provide ongoing feedback that can be used by instructors to improve their teaching and by students to improve their learning.

**Learning Outcomes:** Defining the knowledge, skills and abilities that students should possess following a particular educational experience.

**Problem-Based Learning:** A student-centered pedagogy in which students learn about a subject through the experience of problem solving.

**Self-Directed Learning:** A process by which individuals take the initiative, with or without the assistance of others, in diagnosing their learning needs, goals, resources and strategies.

**Software Engineering:** The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.
ENDNOTES

1  http://www.computer.org/portal/Web/sWerbok

2  http://www.youtube.com/watch?v=-xrrk-XhgVc

3  Direct quotes from student and lecturer feedback are presented in italics within the text.

4  http://www.youtube.com/watch?v=4K39UWxdm0U