Introducing Problem-Based Learning in a Joint Masters Degree: Offshoring Information Technologies

Vincent Ribaud, Philippe Saliou

Introducing Problem-Based Learning in a Joint Masters Degree: Offshoring Information Technologies

Vincent Ribaud, Philippe Saliou
Laboratoire en Sciences et Techniques de l'Information
Université de Brest, UEB, LabSTICC
Brest, France
{ribaud, psaliou}@univ-brest.fr

***Abstract*— A young offshore software industry has grown up in Morocco. The University of Brest has set up a network of major software companies and Moroccan universities, providing two mobility schemes towards France. Both schemes include a final internship on the French side of global companies, with pre-employment on the Moroccan side – a successful internship being the key that opens the door to recruitment. Student heterogeneity, and student reluctance to move towards a professional attitude are important barriers to employability. Hence, we redesigned a significant proportion of our technical courses to use a problem-based learning (PBL) approach. The PBL approach is illustrated through drawing parallels with the production of a TV series. Three aspects of the approach are presented: (i) set-up of the studio in which sessions are run, i.e. a real software project, its work products and its software development environment; (ii) pre-production tasks including the screenwriting of problem-based learning scenarios and the procurement of input artefacts; and (iii) acting, i.e. students' interpretation of characters (roles) and teacher direction.**

***Index Terms*— student employability, global software development, problem-based learning**

I. Introduction

The growth of Global Software Development has impacted the informatics education system, and universities are now offering specialized courses or entire programmes dedicated to Global Software Development / Global Software Engineering (GSD/GSE) [1, 2, 3, 4]. The young Moroccan offshore industry has rapidly grown up as an attempt by French software companies to satisfy their clients' desire to offshore software projects. The Moroccan government has completed several initiatives aimed at fostering offshore industry. With regard to IT education, government funding has helped start new programmes called "Masters in Offshoring " at almost every Moroccan university. In 2007, an informatics teaching network was set up, comprising Moroccan and French universities. Moroccan and French stakeholders agreed to our university's proposal to act as a kind of placement agency providing some students with an internship in France. Ensuring graduates will return to the country of origin (Morocco) was seen as a crucial issue, and one that can only be guaranteed by strong institutional governance of each student's mobility. Recently, we replaced this mobility scheme with the possibility of basing the final year of study in France, leading to the award of a double Masters degree - Moroccan and French. The whole programme is called Offshoring Information Technologies (*Offshoring des Technologies de l'Information* - *OTI*). The programme involves major industrial players in offshore development: Logica, Capgemini, Atoss – as well as nine Moroccan state universities.

We introduced a Problem-Based Learning (PBL) approach within some of the programme courses, mainly in an attempt to resolve two problems: heterogeneity of knowledge and skills between students, and reluctance on the part of certain students to transition from a passive learning attitude to one that is active. General issues are discussed in section II, and the OTI programme itself is described in Section III. Section IV presents an introduction to PBL, the practicum in which it is run, the screenwriting of problem-based learning scenarios and procurement of input artefacts, and student interpretation of roles directed by teachers. We finish with a brief conclusion.

II. Issues Analysis

A. Governmental issues

In 2008, Gartner Research published a report on the Analysis of Morocco as an Offshore Location [5]. This report pointed out that Morocco is an attractive 'nearshore' alternative for Europe, and that several established companies have nearshore centres in Morocco. They noted also that Morocco has yet to provide a clean and democratic environment, although it is making progress in this area. In order to foster the development of Morocco as an offshore country, the Moroccan government has implemented several initiatives to promote the Information and Communication Technology (ICT) industry - including, in December 2006, an emergence plan entitled "10,000 ingénieurs" (10,000 engineers). This plan aimed to provide the software development market with 10,000 novice engineers per year. Although in 2006, just 4,000 such novices had graduated, by 2010, they numbered 10,600 - including 3,700 Masters graduates issuing from state universities. The

government's current objective is to train 15,000 engineers a year from 2015, and 25,000 from 2020.

B. High education issues

The Mediterranean Office for Youth - MOY (http://www.officemediterraneendelajeunesse.org/en) was recently established in recognition of the fact that circular migration for educational purposes is a decisive factor in the development of wealth, intercultural exchange, and mutual understanding in the Mediterranean region. The MOY is operating in 14 countries around the Mediterranean, and is labelling higher education training programmes of excellence corresponding to fields of Mediterranean interest. The MOY label is awarded to Masters and PhD programmes meeting the conditions and criteria set by MOY for the purposes of facilitating student mobility in disciplines identified as priorities for the development of the Mediterranean region, and promoting the employment of young people in their country of origin. We responded to the first call for proposals for MOY labelling, and our programme - along with 41 others - was selected. It is the only joint Masters in information technologies /software engineering.

C. Companies' issues

The notion of distance is considered a major factor impacting Global Software Development (GSD) [4, 6]. GSD teams are usually made up of members from different countries, speaking different languages and with different managerial traditions. This is called the socio-cultural distance. Almost all initiatives indented to reduce socio-cultural distance rely on a long period of immersion in the foreign culture.

When we started the programme in 2007, the major players in the Moroccan offshore software industry (Logica, AtoS, Capgemini, and HP-CDG) asked us to provide facilities that would enable Moroccan and French team members to spend time together in order to help French and Moroccan teammates "rub up against one another". We made a pragmatic response offering prospective young Moroccan employees the opportunity of a stay in a French company that is long enough to understand how French teams behave, professionally.

D. GSD education programme

Few universities offer entire programmes intended to prepare IT engineers to work in a multicultural environment. Detroit Mercy University has offered such a course for more than 20 years now: International Studies in Software Engineering Program (ISSE). The main course of action is to immerse students in foreign culture - which is also our principal method. Our programme differs in that we offer Moroccan students an experience in a foreign university and in a foreign business (the French side of the company linked with the potential Moroccan employer).

In Europe, we are aware of two European Masters programmes in Global Software Engineering, which are named: European Master on Software Engineering (EMSE, http://emse.fi.upm.es/) and Global Software Engineering European Master (GSEEM, http://www.gseem.eu/). Both of these use a 1-year mobility scheme, with the first year completed at the university of origin and the second at a foreign university. Like our proposal, this is a one-year foreign immersion leading to a double Masters degree. Both programmes are research-oriented. Compared to existing programmes, the most distinctive feature of our programme lies in its strong career orientation, since it is designed to gain an initial professional experience in France that is intended to lead to employment in Morocco.

III. Description of the Programme

A. Fundamental Principles

Professional integration issues have been at the heart of the programme ever since it was started, back in 2007. Strict control of mobility is required. The French government's priority is to prevent illegal immigration, while Morocco wants to hang on to its most talented people. The partners have therefore agreed:

1) A founding principle: Acquire a first experience in France and then mobilize the skills gained, for the benefit of Morocco's economic development.

2) Centralized co-ordination of mobility and employability: This co-ordination is supported by the University of Brest, which acts as a hub connecting Moroccan universities, Moroccan students, future Moroccan employers and French companies working in offshore software development. The university also co-ordinates the various academic, administrative and legal procedures.

B. Terms of mobility

The OTI programme includes two mobility schemes. Since 2007-2008, the scheme called "Stage en France avec une pré-embauche au Maroc" (SFM), *Internship in France with pre-employment in Morocco*, provides mobility over one semester. In 2010, we replaced this scheme with another, based on mobility over one year. This is a joint Masters degree from the University of Brest and any one of 9 Moroccan universities. The first year of study takes place in Morocco, the second in France: 6 months of study at Brest, followed by a period of 6 months in France, with pre-employment in Morocco.

Both mobility schemes use internship as a placement mechanism. All stakeholders share a single goal: the recruitment of Masters graduates. French companies' expectations of Moroccan interns are high, especially since they are considered to be (and indeed are) normal French Masters graduating students. For almost all Moroccan students, this internship in France is their first encounter with the industrial world and its expectations. Some interns experience difficulty in adopting a professional attitude and in leaving their student clothes at home - literally or figuratively. We have the same problem on a five-year curriculum in Computer Science, where there is just one, final internship: moving towards the job market is difficult for most students. Preparing students for the real world was one of the main reasons behind the introduction of the PBL experience for Moroccan students.

C. Statistical data

While the initial Moroccan partners followed a common curriculum framework (called Masters in Offshoring), the first year of the Masters in Morocco can now be performed in four quite different specialties:

- Software development and quality: Hassan II Mohammedia (UH2M-Casablanca), Chouaïb Doukkali (UCD-El Jadida), Sidi Mohamed Ben Abdellah (SMBA-Fès) and Ibn Tofaïl (UIT-Kenitra) universities;
- Networking and Systems: Ibn Zohr (UIZ-Agadir), Hassan II Mohammedia (UH2M-Casablanca), Hassan 1er (UH1-Settat) and Abdelmalek Essaâdi (UAE-Tanger) universities;
- Information System Engineering: Cadi Ayyad (UCAM-Marrakech) university;
- Applied Informatics Offshoring: Mohammed V-Agdal (UM5A-Rabat) university.

1) *Students' origin:* Table I shows the number of double Masters students for whom the University of Brest was responsible between 2010 and 2013 (the current year).

TABLE I. MASTERS' STUDENTS COUNT BY UNIVERSITY OF ORIGIN

University	10-11	11-12	12-13	Σ
Agadir (Ibn Zohr)	5	5	1	**11**
Casablanca (Hassan II Mohammedia)	9	8	5	**22**
El Jadida (Chouaïb Doukkali)	-	3	2	**5**
Fès (Sidi Mohamed Ben Abdellah)	-	-	5	**5**
Kenitra (Ibn Tofaïl)	9	5	3	**17**
Marrakech (Cadi Ayyad)	2	4	4	**10**
Rabat (Mohammed V-Agdal)	6	6	5	**17**
Settat (Hassan 1er)	-	2	-	**2**
Tanger (Abdelmalek Essaâdi)	-	2	2	**4**
Total	**31**	**35**	**27**	**93**

2) *Employment:* The cumulated counts of both mobility patterns give the hiring rate at the end of the internship. Table II presents the percentage of interns kept on at their companies following the internship. The overall percentage is 103 interns employed over 143 internships, i.e. a hiring rate of 72%. But it may be that student reluctance to move towards a professional attitude is an important barrier to employability, the issue that first led us to introduce the PBL approach.

TABLE II. EMPLOYMENT COUNT AFTER INTERNSHIPS

Company	08	08	09	09	10	10	11	11	12	12	Σ
	Int.	**Hire**	**Int.**	**Hire**	**Int.**	**Hire**	**Int.**	**Hire**	**Int.**	**Hire**	
AtoS	3	3	2	2	-	-	6	3	2	1	**69%**
Capgémini	6	6	13	10	-	-	-	-	3	3	**86%**
HP-CDG	2	2	-	-	-	-	-	-	-	-	-
Logica	3	3	12	6	20	19	41	30	30	15	**68%**
Total	**14**	**14**	**27**	**18**	**20**	**19**	**47**	**33**	**35**	**19**	**72%**

D. Content of the double Masters degree

The knowledge base acquired by the end of the first year may vary from student to student, raising a problem of heterogeneity – and this was the second reason for deciding to try out the PBL approach reported in this paper.

From September to March in the second year of the Masters, all students attend 8 technical courses: Database and Java Programming, Development Environments, Object-Oriented Design, Distributed Systems, Web Technologies, Software Engineering, Information Systems, and J2EE Development. They also attend courses in English and Communication in French, and a course providing a general introduction to offshore context. The 6-month internship takes place from April to September. The programme curriculum has been designed to train engineers in the development (design, production and maintenance) of software projects, rather than just focusing the curriculum (as other GSD courses or programmes do) on offshore-specific aspects. The programme objective is to acquire a foundation of skills and knowledge on the new technologies and industrialization tools used in large software development companies. It is assumed that the processes, methods, techniques and tools of offshore development vary from company to company and are taught and mastered during the training internship, which should also be a formative period.

IV. PROBLEM-BASED LEARNING

A. Introduction

Boud [7] introduces his book on Problem-Based Learning with: "*PBL is a way of constructing and teaching courses using problems as the stimulus and focus for student activity. [...] It is a way of conceiving of the curriculum as being centred upon key problems in professional practice.*"

We have experience of applying PBL to the entire final year of a Software Engineering Masters degree [8]. We decided to infuse PBL in three courses. The selected courses are: Database and Java Programming (48h), Software Engineering (60h), and Information Systems (60h): a total of 168 hours – one third of the technical courses as a whole. They are taught by three professors, including both authors of this paper.

PBL is performed through PBL sessions. The PBL approach raises several issues that can be illustrated using the production of a TV series as a metaphor; when someone decides to create a new series, she develops the show's elements – namely, concept, characters, crew, and cast.

The concept and characters yield the background of each PBL episode. The concept of the PBL series is the maintenance and the development of an information system (IS). The characters are the representation of the different jobs that are involved in maintaining and developing an IS.

The crew is a group of people in charge of producing the PBL series. Crew are distinguished from cast, the actors. The crew is divided into different sectors, each of which specializes in a specific aspect of production. Some crew positions will be highlighted in this paper. Crew members are academics.

The cast consists of the actors who appear in front of the camera or provide voices for characters in the film. Actors are students. They have to learn, and portray, their characters.

In the film industry, the main production phases are pre-production, principal photography, and post-production. Pre-production begins when a script is approved. Pre-production tasks include storyboarding, construction of sets, props, and costumes, casting, budgeting, acquiring resources, etc.

Principal photography is the actual filming of the episode, where people gather at a television studio or on location to film the scenes of the episode. Once principal photography is complete, the producers co-ordinate post-production tasks.

In our PBL production, pre-production consists of all tasks required to prepare the PBL session, including script writing - a major task. Since the purpose of the sessions is not to record episodes for broadcasting, but to focus instead on the role play, we will call this phase Enacting. We do not have post-production tasks.

B. The practicum

The concept runs through the series, and in our case, concerns the development of information systems through successive phases performed by specialized characters who must stay in role. Concept and characters are set up in a practicum: all together on the sets where the sessions are performed, the decors used in each session, and the accessories required for interpretation of the characters.

1) *Architecture:* A Management Information System, called SIGILI, has been developed to meet the needs of our Informatics Department. SIGILI was designed to manage schooling and was used by administrative staff and programme managers. SIGILI is composed of 3 sub-systems:

- SIGILI1, a schooling management system;
- SIGILI2g, an internships management system;
- eCompas, a competencies management system.

The whole system was developed between 2005 and 2007 with the second author acting as project manager (the job he used to perform at software companies for 13 years prior to joining our university); each sub-system was developed by a team of 5-6 full-time interns during their 7-month Masters internship (17 interns in all). The three sub-systems use a 3-tier architecture in which the user interface, functional process logic, computer data storage and data access are developed and maintained as independent modules, on separate platforms. SIGILI1, the first sub-system, was developed with open-source tools and uses Eclipse/Struts as a development framework, and Tomcat as an application server. SIGILI2g and eCompas both use JDeveloper and ADF Faces as an application development framework and the Oracle Application Server. Oracle is used as the DataBase Management System (DBMS) in all three cases.

2) Legacy, complexity and heterogeneity: A major challenge for IT students is dealing with the complexity and heterogeneity of legacy systems. Information systems are built through successive projects, with people, processes and technologies changing over time. A typical banking or insurance information system includes sub-systems and components produced over a period of 30 years. "*Problem-based learning can help students to learn with complexity, to see that there are no straightforward answers to problem scenarios, but that learning and life take place in contexts, contexts which affect the kinds of solutions that are available and possible* [9]." The SIGILI Management Information System and its technical environment will be used throughout all PBL sessions. The SIGILI data model is – like any IS - fairly complex: 90 tables, 60 views, 50 packages, 600 triggers, and 270 indexes. SIGILI code is managed within several configuration software components. The SIGILI infrastructure relies on different technologies. This complex, heterogeneous, legacy environment is the practicum in which PBL sessions run - a software studio corresponding to studio facilities that are used to make episodes of a series.

3) SIGILI artefacts: As mentioned in previous sections, a key component of the practicum is the SIGILI Information System. Although the work has been done by interns led by an experienced project manager, the project manager has never accepted weak deliverables - because the priority was not the project but rather the internship learning outcomes. Moreover, since the major objective of the internships was the learning-by-doing of software engineering processes, the development cycle was performed with a rigor that might not be matched in real software companies, resulting in an exhaustive set of major deliverables issued in a software project at our disposal. Obviously, the purpose of these project artefacts was not to serve the PBL approach - which we built only recently - and most of these need reworking before they can be used in a PBL setting. SIGILI artefacts are part of the furnishing required to run PBL sessions and form a set that is comparable to a film set (decor and props used in a film).

C. Pre-production tasks

An important job in the pre-production crew is - in our opinion – that of the scenario writer. Savin-Baden and Howell Major [10] conclude a chapter on curricula models with "*In problem-based curricula the problem scenarios should serve as the central component of each module [...] the starting point should be a set of problem situations that will equip students to become independent inquirers [...] and perceive that there are also other valid ways of seeing things besides their own perspective*."

A PBL session should be run according a scenario that is intended to be interpreted on the basis of student performance, rather than serving as a "finished product". From our experience, a PBL session works well when a story is told and when students feel themselves involved in the story. "*Storytelling is one of the most powerful techniques we have as humans to communicate and motivate* [11]." Hence, the writing of PBL scenarios is an activity very close to screenwriting - and PBL session designers act as screenwriters and are responsible for researching the problem and its story, developing the narrative, writing the screenplay, and delivering it, in the required format, to the PBL tutors.

Screenwriting theories help writers approach screenplay by systematizing the structure (Goldman's famous quote "Screenplays are structure" [12]), goals and techniques of writing a script. In the three acts paradigm, act I is the setup (location and characters), act II is the confrontation (with an obstacle), and act III resolution (culminating in a climax and a dénouement). Field [13] preached the three-act structure at 1/4 – 1/2 – 1/4 proportions, built around page-number-specific

turning points. 1/4 - 1/4 - 1/2 proportions are more appropriate to the case of problem-based learning. In a 4-hour session, one hour will be devoted to understanding the setup, then students will spend one hour getting to grips with the problem and tackling obstacles, and resolution will take more than two hours.

1) Problem design: Curricular content must be organized around problem scenarios rather than subjects or disciplines. One key aspect for designers is that we just have to accept the amount of curriculum knowledge that will be taught and learnt, without allowing resentment about this to get in our way. The complexity of problem design is a challenge to many tutors implementing problem-based learning. Relying on previous experience, each author designs their own problems.

2) *Development cycle:* The plot of the PBL sessions concerns the maintenance or development of an information system. Practical understanding of the development cycle of an IS is an underlying objective of any PBL session. PBL sessions can be grouped in logical units, each related to a phase of the development cycle: maintenance, coding, design, etc. It will gradually be revealed to students that each PBL session is contributing to some extent to the development of a new sub-system. Our development approach relies on a waterfall process: requirement capture, requirement analysis, design, implementation. Like most information systems, we are using a systemic method. First, data and processing have to be separately modelled, and then coupled to constitute a unique and integrated system. The building of the system moves through different abstraction levels: statement of work, requirements, design and implementation.

3) *Artefacts:* software development activities rely on work products, called artefacts, either as inputs or as outputs. PBL scenarios are played out within software development phases where output artefacts of one phase are used as input artefacts for the next. Successive cases should rely on sound and complete artefacts (even though they should, ideally, have been produced by students). But it might happen that students have been unable to solve the problem and produce strong artefacts – so that their weak artefacts have to be replaced with strong products. Hence PBL designers have themselves to produce good artefacts to accompany the case; otherwise tutors will find themselves unable to run successive cases with students. To understand the burden of this task, recall that an episode (a PBL session) will go through successive scenes, each scene requiring a different film set, which includes the furnishings and all the other objects that will be seen in the scene. For each scene of the PBL session, a new artefacts set is required. Unfortunately, in most cases, the artefacts have to be built by the PBL designer, acting as head carpenter, set decorator and prop maker, to use film industry terminology.

4) *Inverting the cycle:* project-based approaches have to follow the development cycle along its normal path: from requirements to code. During a project, students are supposed to learn the different phases according to the waterfall schedule. Unfortunately, teaching and learning are much more difficult in the uphill phases than they are in the downhill phases. Nobody will try to learn to ski at the top of a mountain where the runs are vertical; instead we learn where the slope is gentle and gradually move up. Applied to software engineering, this means that students are passing through an inverted cycle; the first sequence of PBL sessions is intended to master the implementation activity (from design to code); then a steeper segment is envisaged: the design activity (from requirement analysis to technical solution design); and the last PBL sessions sequence is devoted to requirements analysis, the steepest part of the cycle.

D. Enacting the PBL sessions

Bear in mind that courses chosen for PBL are centred on the development (in the broad sense of software engineering: from requirements to implementation) of information systems.

In a systemic development method, the first phases aim to reach sufficient consensus on problem understanding to produce, as a basis for the next phases, a conceptual data model (as an E-R schema or an UML class diagram) and a conceptual processing model (such as a use case diagram or function hierarchy). Based on this broad understanding, data and process modelling may, to some extent, be separately modelled. Later on in the cycle, data and processing implementation will be coupled again and tightly integrated. Since the PBL sessions are focused on the left branch of the V-model, we gather the PBL sessions into a 3 topic-breakdown: information system engineering (data & processing), database server (data), and application server (processing).

Apart from the development cycle dimension, systemic methods also consider a dimension using an abstract-concrete axis in which models transition from abstract representations to concrete constructs with three different levels: conceptual, logical and physical. Broadly speaking, the "information system engineering" topic focuses on conceptual and logical levels, while "database server" and "application server" topics focus on the physical level. As mentioned in the previous section, we have inverted the development cycle so that the first PBL period is related to physical models.

1) *Database server:* due to student diversity, very few pre-requisites are required, mainly a knowledge of SQL DDL and DML. The first sessions are intended to improve student familiarity with real-scale database schemas. Examples of PBL sessions are:

- Restructuring a set of Data Definition Language (DDL) scripts in a design-based hierarchy
- Checking consistency between code artefacts and technical detailed specification
- Refactoring the DDL sources of a complete sub-system according to naming and organization rules

Once a practical understanding of what a real-scale physical data model is, the next step is to train students in data implementation activities, i.e. transforming a logical model into DDL constructs. It should be pointed out that an understanding of this transformation is obviously key not only to successful implementation but also to successful design. Hence, our approach is to perform a retro-design of the DDL

sources prior to the implementation itself. In the educational field, retro-engineering is an inductive approach. It is the reconstruction of a process from back to front, having the result of an activity as its starting point. Examples of PBL sessions are:

- Retro-designing a set of DDL sources (the physical model) in a logical model
- Producing (mostly generating) the DDL sources again from the logical model and drawing up the logical model and iterating the generation process until the logical model can serve as a reference for the development of the data server side of a complete sub-system.

2) Application server: once again, due to student diversity, very few pre-requisites are required, mainly a knowledge of Java. The first sessions are intended to familiarize students with the application development environment (JDeveloper / ADF Faces). Examples of PBL sessions are:

- Running a step-by-step tutorial, then applying it to programming a small software component having a similar structure
- Retro-designing then developing a set of Web pages with the user's manual yielded as specifications
- Performing a code review on an existing module

Obviously, with a complex development framework such as Eclipse / Struts or JDeveloper / ADF Faces, a long learning curve is inevitable, and such PBL sessions are only intended to prepare students to implement either interactive or batch processing functions from a design specification. Unfortunately, programming tools do not provide the same maturity as data modelling tools, and there is no substitute for programming experience. PBL sessions are similar to typical programming labs, except in that they take place in a real system and can be related to the database server PBL sessions. To assist students in using a complex development environment, some areas of PBL lessons are formulated as a tutorial, scaffolding students if necessary. Examples of PBL sessions are:

- Integrating existing pieces of code in a bottom-up approach
- Examining the gap between the solutions provided in a tutorial and expected implementation
- Finally, developing - in a traditional fashion - the code of a sub-system component

3) *Information system engineering:* as mentioned before, database server programming and application server programming were performed in relatively-independent PBL sessions, and focused on the physical levels. We now reach the uphill phases: requirement analysis and software design - and deals with conceptual and logical models or logical models only. Both data and processing functions are modelled. At the time of writing, an initial PBL period of 8 weeks has been completed and reported on in this paper. During the upcoming period, PBL will be applied to analysis and design. We will continue to climb the mountain from bottom to top, learning software design before requirements analysis. The inductive approach will be used: from detailed design to architecture, from architecture to requirements. The last sequence of PBL sessions will be devoted – at last – to performing the uphill phases in the usual, top-down fashion – from requirements to architecture, from architecture to detailed design, from detailed design to implementation – with the practical knowledge and skills gained during the inductive PBL sessions.

E. Students' and teachers' role

Active learning refers to several education paradigms that focus the responsibility of learning, on learners. PBL is one active learning method that follows a constructivist perspective in learning. Constructivism can be summed up in two fundamental statements [14]: (i) learning is defined as an active process for knowledge building rather than a knowledge acquisition process; (ii) teaching is essentially aimed at helping students in this process rather than transmitting knowledge.

Among practices belonging to the constructivist stream (and cognitive psychology), D. Dwyer [15] and J. Tardif [16] define a learning paradigm, in opposition to the main teaching paradigm.

1) *Teachers' roles:* J. Tardif defines teachers' roles as creators of pedagogical environments; interdependent, open-minded, critical professionals; development instigators; mediators between knowledge and students; coaches; collaborators for the student success of a whole school.

As mentioned in [17] "*In many universities, the adoption of problem-based learning is adding another dimension to what it means to be a lecturer in higher education."* Among the roles mentioned above, we emphasize the roles of creating pedagogical environments for the PBL sessions and of coaching whilst PBL sessions are running. Both authors feel they have a lot to learn themselves about the job of being PBL coaches (called PBL facilitators in the literature). We lack support from the university for those staff who are Problem-Based Learning facilitators. We also have little understanding of the complex interactions between team and facilitator during the PBL - and how both sides adapt their behaviour as PBL practice matures.

2) *Student roles:* J. Tardif defines student roles as investigators; co-operators sometimes experts; clarifying actors; strategic users of available resources. Among the roles mentioned above, the investigator and strategic user roles are most important.

PBL research is usually enthusiastic about PBL adequacy and effectiveness applied to engineering and medical science. For instance, [18] claimed that "*Student learning changed and student knowledge increased as a result of implementing PBL*." Satisfied students report the same viewpoint. But, as pointed out by Boud [7]: "*The principal idea behind problem-based learning is [...] that the starting point for learning should be a problem, a query or a puzzle that the learner wishes to solve.*" But it can happen that some (or all) students do not wish (or are unable) to solve a problem. Another point is that students do notice when, for one reason or another (inadequate preparation, lack of experience of the PBL tutor, weaknesses in the inputs

artefacts provided, etc.) a PBL session fails to work. Unless students sign up to the PBL approach, they might use the failed lessons to weaken the approach. Sweenev [19] clearly pointed out that the PBL concept should be clear to all and that everybody should understand PBL to mean the same thing, otherwise it may frequently induce discomfort, confusion, antipathy, lack of co-operation and general disbelief in PBL.

F. Assessment

If we consider the SWEBOK topics addressed in the PBL series (http://www.computer.org/portal/web/swebok), they belong to three Knowledge Areas (KA): software requirements, software design, software construction. Annex D of SWEBOK presents a classification of KA topics according to Bloom's taxonomy: Knowledge (K), Comprehension (C), Application (AP), Analysis (AN), Synthesis (S), Evaluation (E). We consider the scope of PBL sessions and mention the topics addressed within the sessions together with the associated Bloom level in brackets:

- SW requirements sessions are focused on requirements classification (AP), conceptual modelling (AN), architectural design and requirements allocation (AN), software requirements specification (AP)
- SW design sessions are focused on architectural structure and viewpoints (AP), structured design (AP), object-oriented design (AN)
- SW construction sessions are focused on construction design (AN), construction language (AP), coding (AN)

All topics are AP-classified (action verbs: apply, change, construct, manipulate, operate, produce, solve, use, ...) or AN-classified (analyse, compare, deconstruct, identify, illustrate, infer, outline, select, ...). Obviously, assessment cannot be performed in the same manner as usual. PBL assessment is part of the PBL itself, as is true of almost all active teaching approaches - what J. Tardif [16] calls “the entrenchment of assessment in learning”.

Our assessment relies essentially on portfolio assessment. When a PBL session artefact is delivered, the tutor examines it and provides feedback about certain points to be improved upon or started over. Ideally, feedback is given in front of the authors, allowing the authors to delve deeper, discuss, and even contest remarks made by the tutor. But the workload may be too heavy and we also practice a stop-and-go approach: it works or it does not work. In the latter case, students are poorly assessed but are provided with a working artefact that allows them to continue their work.

Formal examinations take place every two months; this was therefore the case at the end of the fully-PBL period. Examinations are based on work performed by students during the session, and may be considered as PBL sessions themselves - though without any help from tutors.

PERSPECTIVES AND CONCLUSION

This article presented the introduction of a PBL approach in a mobility programme for Moroccan students coming to France, governed by a strong principle of directing skills for the benefit of Moroccan economic development. However, student heterogeneity and lack of industrial experience confronted us with new challenges, hence the PBL approach was trialled on a few courses in order to develop a reflective practice.

Although PBL has proved its worth in engineering education, an immediate conclusion is that the price of starting a PBL approach is high, a drawback to bear in mind. Our experience is too limited to draw any conclusions about student perception of PBL or the pros and cons of the approach. We plan to relate student participation in PBL with their involvement in problem-solving during the internship - one of the fifth assessment indicators used in awarding a mark to the internship.

REFERENCES

[1] C. Deiters, C. Herrmann, R. Hildebrandt, E. Knauss, M. Kuhrmann, A. Rausch, B. Rumpe, K. Schneider, "GloSE-Lab: Teaching Global Software Engineering," 6th IEEE Int. Conf. on Global Software Engineering, pp.156-160, Aug. 2011.

[2] D. Petkovic, R. Todtenhoefer, G. Thompson, "Teaching Practical Software Engineering and Global Software Engineering: Case Study and Recommendations," 36th Annual Frontiers in Education Conference, pp.19-24, Oct. 2006.

[3] N. R. Mead, D. Shoemaker, A. Drommi, J. Ingalsbe, "An Immersion Program to Help Students Understand the Impact of Cross Cultural Differences in Software Engineering Work," 32nd IEEE Int. Conf. Computer on Software and Applications, pp.455-459, July-Aug. 2008.

[4] P. Lago, H. Muccini, L. Beus-Dukic, I. Crnkovic, S. Punnekkat, H. Van Vliet, H.; , "Towards a European Master Programme on Global Software Engineering," 20th Int. Conf. on Software Engineering Education & Training, pp.184-194, July 2007.

[5] S. Karlsson and I. Marriott, *Analysis of Morocco as an Offshore Location*, ID Number: G00161819, Gartner Publications, 2008.

[6] M. J. Monasor, A. Vizcaíno, M. Piattini, I. Caballero, "Preparing Students and Engineers for Global Software Development: A Systematic Review," 5th IEEE Int. Conf. on Global Software Engineering, pp.177-186, Aug. 2010.

[7] D. Boud and G. Feletti, “Changing problem-based learning [Introduction].” In D. Boud & G. Feletti (Eds.), *The challenge of problem-based learning* (2nd ed). London: Kogan Page, 1997.

[8] V. Ribaud and P. Saliou, "A project-based immersion system" In 21st IEEE-CS Conference on Software Engineering Education and Training Workshop (CSEETW '08), pp.25-28, April 2008.

[9] M. Savin-Baden, *Problem-based Learning In Higher Education: Untold Stories*, Maidenhead: Open University Press- McGraw-Hill Education, 2000.

[10] M. Savin-Baden and C. Howell Major, *Foundations of Problem Based Learning*, Maidenhead: Open University Press- McGraw-Hill Education, 2004.

[11] L. Widrich, " What listening to a story does to our brains", http://blog.bufferapp.com/science-of-storytelling-why-telling-a-story-is-the-most-powerful-way-to-activate-our-brains (last visited, 2012, December 1th.

[12] W. Goldman, Adventures in the Screen Trade: A Personal View of Hollywood and Screenwriting, 1983, NY: Warner Books.

[13] S. Field, Screenplay: The Foundations of Screenwriting, 2005, New York: Delta.

[14] M. Duffy and D. J. Cunningham, "Constructivism : Implications for the design and delivery of instruction", In Handbook of

Research for Educational Communications and Technology, London: MacMillan, 1996.

[15] D. Dwyer, Apple Classrooms of Tomorrow : What we have learned, In Educational Leadership, vol. 54, num. 7, 1994.

[16] Jacques Tardif, Intégrer les nouvelles technologies de l'information – Quel cadre pédagogique ?, Paris: ESF, 1998.

[17] M. Savin-Baden, *Facilitating Problem-based Learning: Illuminatiing Perspectives*, Maidenhead: Open University Press-McGraw-Hill Education, 2003.

[18] I. Richardson and Y. Delaney, "Problem Based Learning in the Software Engineering Classroom," 22nd Conf. on Software Engineering Education and Training, pp.174-181, 2009.

[19] G. Sweenev, "The challenge for basic science education in problem-based medical curricula", Clinical and Investigative Medicine, 22, pp. 15-22, 1999.