yPBL methodology: a problem-based learning method applied to Software Engineering

Ernesto Exposito
CNRS ; LAAS ; 7 av. du Colonel Roche, F-31077 Toulouse, France
Université de Toulouse; UPS, INSA, INP, ISAE; LAAS; F-31077 Toulouse, France
ernesto.exposito@insa-toulouse.fr

ABSTRACT

This paper proposes the yPBL learning methodology, based on the well-known PBL method and adapted to software engineering process by using the "y" methodology. The yPBL methodology is defined as a mapping between the roles and phases considered in PBL methodologies to the roles, iterations and phases considered in the "y" methodology. Moreover, the yPBL method includes different situations of active and passive learning roles not only for the students involved in a course but also for the instructors. Indeed, software engineering instructors face the same challenge of any software engineer and needs to continuously update their knowledge in software technologies. The yPBL method has been designed using the Unified Modeling Language (UML) and the various interactions points between the various process actors as well as the information to be exchanged during the synchronous and asynchronous learning process have been specified using this language. Finally, interesting preliminary results of the experience of using this methodology in the INSA of Toulouse are included in this paper.

Keywords-software development process, problem based learning, unified modeling language, software engineering process

I. INTRODUCTION

Software engineering is a complex process demanding from development team members a high level of knowledge and experience in diverse areas going from project management skills to communication, design and implementation expertise. Moreover, the large diversity of software design and development approaches as well as the accelerated development of new software technologies requires a continuous learning process. This is not only the case for software engineers but also for academic instructors teaching software design and development courses.

Problem based learning (PBL) methods have been successfully used in different domains and its benefits have been largely demonstrated [1,2]. These methodologies ask for the active participation of the students within the learning process, playing not only the traditional passive learning role but also an active role where part of the knowledge needs to be discovered and applied by themselves. Moreover, the students may be asked to transmit the knowledge they have acquired to other students in order to reinforce the learning process as well as to demonstrate that the learning objectives have been achieved. Nevertheless, even if PBL methodologies have been designed to be easily adapted to any educational domain, the specificities of software engineering courses need to be carefully studied in order to improve the benefits of these learning methods while applying good practice approaches that are specific to this domain. As previously introduced, software engineering courses ask for multiple skills expertise acquisition and development. Indeed, students need to efficiently exert in the area of project planning, quality assurance, translation and traceability of customer requirements, analysis of the software context and constraints, mapping of functional and non-functional requirements to technical requirements, design of software solutions following good and well-known practices (e.g. design patterns or object oriented approaches), implementation of the designed solution, testing techniques, integration procedures and finally deployment and maintenance strategies of the software product and related documentation. The previous list is not exhaustive and shows the degree of complexity involved in designing PBL-based courses allowing students to play the required roles in order to achieve the software engineering learning objectives.

In the area of software engineering process, several methodologies have been proposed in order to efficiently support members of development teams to design and implement software products. Unified Process (UP) methodologies are very well known in the world of software engineering for providing an efficient process based on an incremental and iterative sequence of phases. Phases include analysis and specification of requirements, design and specification of the software solution and implementation, test, integration and deployment of the software product. These phases are planned and executed in incremental iterations where in each increment new customer requirements can be added within the process. Likewise, bugs detection and corrections as well as requirements change requests can be added in each iteration. As agreed in the software management plan, stable or experimental software products can be released at the end of the iterations.

UP methodologies have been specialized in the form of extended methodologies (i.e. Rational Unified Process, Enterprise Unified Process, Extreme Unified Process, Agile Unified Process, etc.). In recent years an interesting specialization known under the name of Two Tracks Unified Process (2TUP) has been proposed to face the reality of continuous changes of requirements and technologies that represents an invariant reality in software engineering. This methodology, also known as the "y" methodology due to its graphical representation, proposes a differentiation of 2 tracks for the Unified Process, the first (left) track represents the functional aspects of the software product and the second

978-1-4244-6571-2/10/$26.00 ©2010 IEEE

(right) track the technical aspects (e.g. technology, environment, platforms). This separation helps software engineers to concentrate on discovering and specifying the functional requirements that need to be satisfied (left track) while allowing them to explore and select the technologies that could be used to build the software solutions (right track). Once the functional and technical requirements have been identified and specified, both functional and technical tracks can be merged in order to produce the software design specification. From this point, the software product can be developed, tested, integrated and deployed. This sequence of parallel and serialized phases will be executed within the incremental and iterative process proposed by the UP method. Benefits of this interesting methodology have been demonstrated by its application in many industrial and research software projects.

This paper proposes a new learning methodology, based on the well-known PBL method and adapted to software engineering processes. This methodology called yPBL is aimed at being applied to develop software engineering courses within the context of real software projects. yPBL is defined as a mapping between the roles and phases considered in PBL methods into the roles, iterations and phases considered in the "y" process. The yPBL method defines a process where incremental and iterative phases and communication channels and deliverables are planned and defined to facilitate the interaction between external and internal actors involved in the real software project: “the client” and “the project team”. Within the project team, students and instructors work together playing different roles in order to build the software solution required by the client. Guided by a real project, internal actors of the process are naturally involved in situations of passive and active learning. Indeed, similar to the students, software engineering instructors face the same challenge of any software engineer face to the accelerated software engineering evolution. For this reason, both instructors and students need to participate in a continuous learning process. The yPBL methodology also defines an internal process where the interactions between the internal actors are planned in incremental iterative asynchronous basis. In order to perform these interactions, internal actors need to work on learning activities including bibliographic research, course preparation and presentation and evaluation of peers based on real knowledge acquisition. Further in the software building process, internal actors need to apply their acquired knowledge in constructing the software solution. At the end of each planned iteration, interactions with the external actors (i.e. clients) are carried out in order to present and evaluate the product releases. During these interactions, the evaluation of the software product is done based on the client’s satisfaction degree and the product qualities.

The yPBL method has been designed using the Unified Modeling Language (UML) and the various interactions points between the various process actors as well as the information to be exchanged during the learning, software construction and evaluation process have been specified using this language. The yPBL methodology has been successfully applied in several software engineering courses at the INSA of Toulouse.

The rest of this paper is organized as following. Section I presents a state of the art related to software engineering processes. Section II presents the yPBL methodology model. Section III describes a concrete study case illustrating the use of yPBL within a Software Oriented Architectures (SOA) course. Finally, several conclusions and perspectives of this work are presented.

II. STATE OF THE ART

In this chapter the state of the art aimed at providing the basis for the yPBL methodology is introduced. First section is aimed at presenting the main standards in software engineering processes. Second section briefly describes the 2TUP process that is the one promoted by the yPBL methodology. Finally, the main IEEE standards aimed at guiding and documenting software engineering process will also be presented.

A. Software engineering process

A software process defines the steps required to create a software product. One of the most mature and well-known software engineering processes is the Unified Software Development Process or USDP [3]. USDP was introduced as a standard process for creating software products based on the use of the *Unified Modeling Language* (UML).

USDP Introduces the concept of 4Ps: people, project, product and process. *People* working in a software development *project* collaborate within an adequate workflow based on the unified *process* using the common UML notation in order to build and represent the blueprint of the software *product*. The process includes all the activities needed to transform user's requirements into a software system. These activities include *project management, requirements specification, analysis, design, development and testing*.

USDP follows a *component-based approach*. This means that the software system being developed is based on software components interconnected via well-defined interfaces. Likewise, *object oriented design and development approaches* are followed within USDP.

There are three major characteristics differentiating USDP from other approaches:

- *Use-case driven*: the process is driven by the use cases or functionalities offered for each external actor (i.e. clients or any external entity interacting with the system). It means that the process does not consider functionalities that “might be good to have", but it is driven by the realistic usages of the system. In other words, use cases drive all the process phases: requirements, design, implementation and test.
- *Architecture centric*: during the process the software architecture is constantly refined including static and dynamic aspects of the system. It means that the form of the system is built progressively.
- *Iterative and incremental process*: the transformation of user's requirements into the software product is performed within an iterative and incremental process. During this process, the functions and the form of the system are represented by the use cases and the architecture respectively.

Various adaptations to the Unified Process (UP) have been proposed in the last years. These adaptations are based on the category of software system being developed, the organization involved, competence levels of development teams or the project size. Examples of these specializations are Rational Unified Process (RUP), Enterprise Unified Process (EUP), eXtreme Unified Process (XUP) or Agile Unified Process (AUP). However, most of the processes used today for designing and developing software systems are commonly based in the principles proposed by the USDP process. This is the case for the 2TUP process described in the next section.

B. *The "y" or 2TUP process*

As previously introduced, the Two Tracks Unified Process (2TUP) has been proposed to face the reality of the constant change of requirements and technologies of current software systems [4]. The "y" methodology proposes an iterative and incremental process composed by 2 parallels tracks aimed at capturing functional and technical requirements, followed by one centralized design track.

This tracks-oriented structure helps software engineers to concentrate on discovering and specifying the functional requirements that need to be satisfied (left track) while allowing them to explore and select the technologies that could be used to build the software solutions (right track). Once the functional and technical requirements have been identified and specified, both functional and technical tracks can be merged in order to produce the software design specification. From this point, the software product can be developed, tested, integrated and deployed. Further details of the "y" methodology will be presented in the next.

C. *Software engineering standards*

Several standards have been proposed in order to guide and document software engineering process. The most widely used in industry are the standards proposed by the IEEE:

- Software Project Management Plan (SPMP): specifies the structure of software project management plans that are applicable to any type or size of software project [5].
- Software Requirements Specification (SRS): specifies the structure and necessary qualities of software requirements specification documents [6].
- Software Design Description (SDD): proposes the necessary information content and recommendations for software design descriptions [7].
- Software Quality Assurance Plan (SQAP): specifies the format and content of software quality assurance plans [8].
- Software Configuration Management Plan (SCMP): describe the structure and content for a software configuration management applying to the entire life cycle of the software [9].
- Software Test Documentation (STD): this document defines the form of a set of documents for use in defined stages of software testing [10].
- Software Validation & Verification Plan (SVVP): specifies the structure of the validation and verification plan including analysis, evaluation, review, inspection, assessment, and testing of software products and processes [11].

These standards help to express and communicate in an unified way all the information related to the software process.

In the case of software engineering learning methodologies, the introduced software processes and related document standards, provide the basis to define a learning model where the learning objectives can be efficiently achieved while designing and developing a real software project. Next chapter introduces this software engineering learning model.

III. YPBL MODEL

The yPBL is a learning methodology, based on the PBL model and inspired in software engineering processes. As previously introduced, yPBL is aimed at being deployed in the context of software engineering courses based on the construction of a real software system. The yPBL model is defined as a mapping between the roles and phases considered in PBL methods into the roles, iterations and phases considered in the "y" process. As an intent to formally describe the yPBL methodology, the UML language has been used to build an yPBL model.

1) *yPBL use cases*

In order to follow the good practices for software engineering introduced in the previous chapter, the unified process has been used to model the yPBL methodology itself. As any unified process, the yPBL methodology is use-case driven as illustrated in Figure 1.

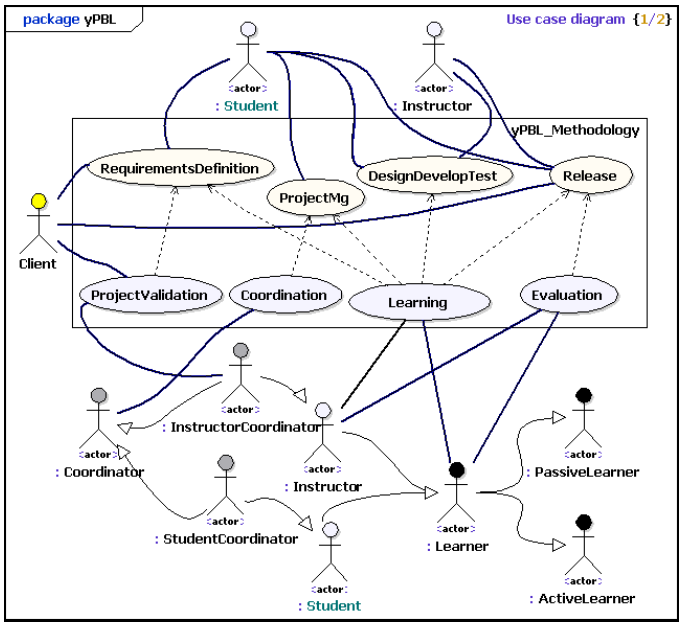

Figure 1. yPBL methodology use cases diagram

In this diagram the various actors interacting in order to achieve the learning objectives while constructing a software system are depicted: students, instructors and the external client.

Guided by the construction of the software project, two generalizations of actor roles are proposed in yPBL: coordinators and learners. The coordinator role is involved in the learning project management and the learner in the learning activities.

Specializations of the coordinator role are represented by instructor coordinator and student coordinator roles. These actors play a supporting role for activities such as planning, scheduling, hardware and software resources allocation. They monitor and control the project in order to early detect potential problems and work together to find efficient solutions. Specifically, the instructor coordinator actor is the one interacting with the external client in order to study and validate the project to be used to instantiate the methodology.

Generalizations of the learner role are defined by passive and active learners roles. Students and instructors play these learner roles. Actually, they are internal actors of the real process and as a consequence they are naturally involved in situations of passive and active learning face to the requirements for learning and applying software engineering technologies. Active and passive learning roles facilitate both instructors and students participating in the continuous learning process.

2) yPBL high level process

The yPBL method follows also the incremental and iterative process proposed by the Unified Process as illustrated in the activity diagram presented in Figure 2.

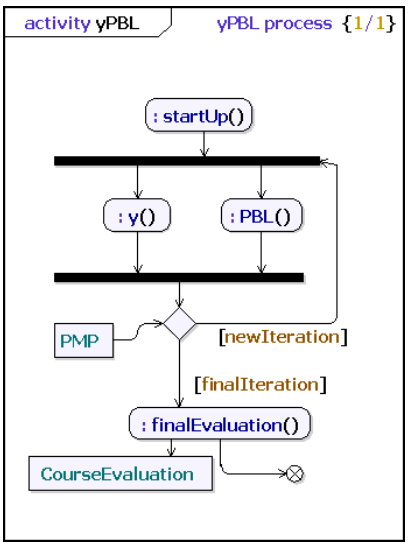

Figure 2. Process proposed by the yPBL

At the high-level yPBL process, an initial start up activity needs to be performed in order to prepare and validate the overall software-learning project. Once the start up activity is achieved, two parallel processes represented by the y and the PBL process are performed. The y process itself concentrates in software engineering activities. The PBL process targets the learning activities. During the overall yPBL process, specific adaptations to the standards presented in the previous section and proposed to guide a software project will be used to drive both y and PBL processes. The first standard is the Project Management Plan (PMP). This document is an adaptation of [5], [8] and [9] documents, and it is intended to control and manage the yPBL process. As illustrated in Figure 2, the PMP document is used for each iteration in order to control the project progress according the initial plan as well as to manage people, resources and deliverables involved in each phase for both software and learning project processes. Once the final iteration has been achieved, a final evaluation of the yPBL project will be performed. During this evaluation both learners participants as well as the learning process itself are evaluated. Results of these evaluations are reported in the "Course Evaluation" deliverable.

3) yPBL detailed level process

The various activities illustrated in the yPBL process presented in the previous section will be further detailed in this section. Specifications used to model internal yPBL activities are intended to describe the workflow process. In these specifications sequence of activates, interaction between the various process actors, as well as communication channels are specified.

Figure 3 illustrates the start up activity. This initial activity is performed as an interaction between the client and the instructor-coordinator.

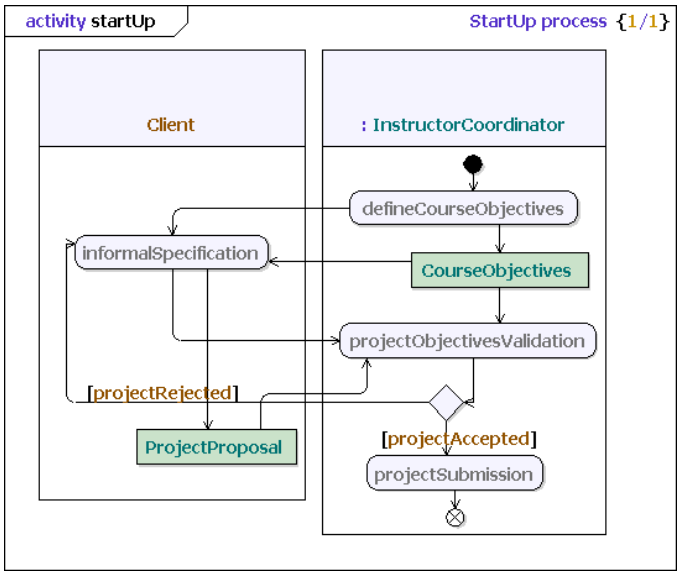

Figure 3. Starting up process

The start up activity starts when the instructor-coordinator defines the course objectives from the functional and technical point of view. Functional objectives are defined as abstract learning statements aimed at expressing the basic and fundamental knowledge goals to be acquired by the learners. Technological learning objectives are intended to express

concrete statements based on current software technologies to be used by the learners in order to apply the basic knowledge goals defined by the functional objectives. The document called "Course Objectives" is used to collect these functional and technical objective specifications. This document is communicated to potential clients in order to allow them to propose an objective-compliant project. Clients are asked to propose an informal specification of the project in the form of a "Project Proposal" that will be validated or rejected by the instructor-coordinator. If the project is accepted it will be submitted to the rest of the yPBL process actors.

From this point and as illustrated by the high level yPBL process in Figure 2, two parallel processes guiding the software and learning project activities are started. Figure 4 illustrates the activity diagram modeling the software project process. Students and instructors perform collaborative or individual activities for every iteration of the y process. In order to stimulate autonomy skills, students are asked to work on the functional and technical analysis phase of the project based on the "Project Proposal" submitted by the client. During this phase, students need to interact with the client in order to clearly specify and validate the software requirements and produce the SRS document [6]. Likewise, students are asked to work on the PMP document in order to define the plan to be executed within the several process iterations. Furthermore, they are also asked to pay special attention in defining a realistic project plan based on the priorities of the requirements expressed by the client.

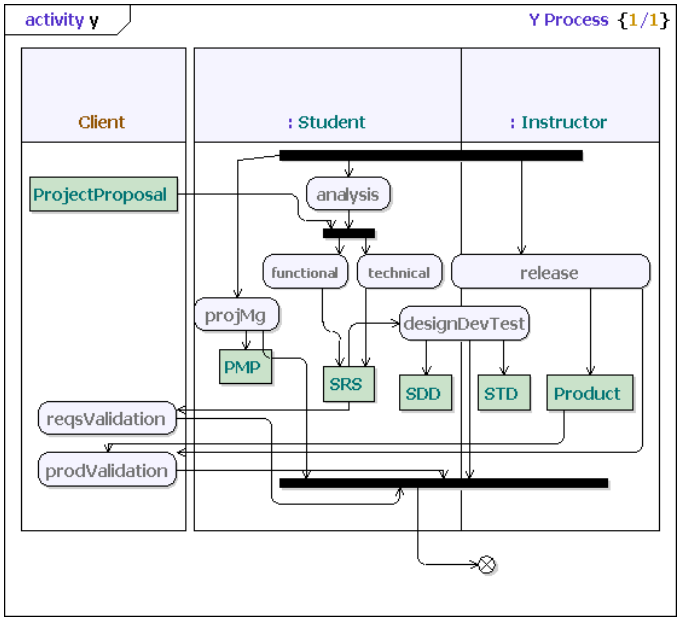

Figure 4. 2TUP or y process

During the design, development and testing activities, both students and instructors work together in order to produce the artifacts expected to be released in every iteration. In these activities the role of the instructor is clearly separated from the client role and this is one important benefit offered by the yPBL method. Indeed, the instructor plays a supporting role intended to help the students to achieve the software project objectives. During these activities, design and test oriented documents are produced following the SDD [7] and STD standards [10]. Following the PMP plan and before the end of the iteration, specific interactions need to be performed with the client in order to validate the "Product" release against the software requirements expressed in the SRS.

In parallel to the y process, learning activities guided by the "Course Objectives" are being carried out for every iteration. Figure 5 depicts the activities performed by the coordinators, instructors and learners during the PBL process.

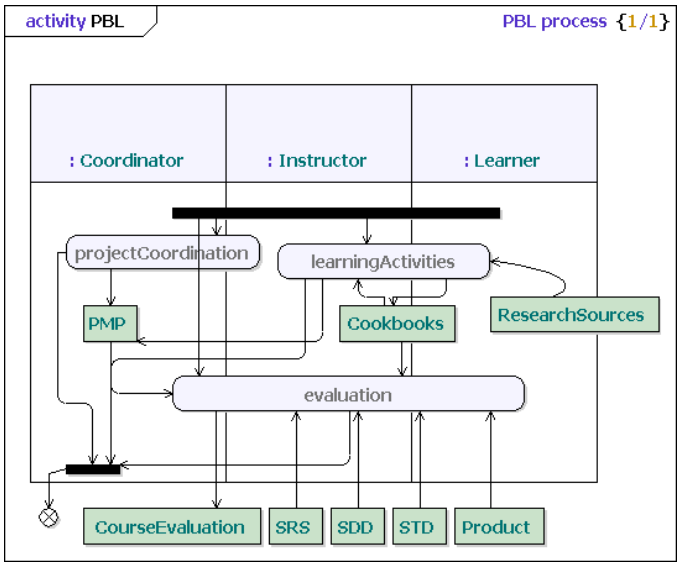

Figure 5. PBL process

Actors performing the role of coordinator (i.e. instructor-coordinators or student-coordinators) perform project coordination activities for every iteration. They work on the PMP document in order to facilitate the project progress and anticipate actions aimed at minimizing potential risks. Coordinators work together on the basis of periodic meetings or by email communication. During these interactions, coordinators exchange monitoring information collected during the process. This information can be used to encourage positive experiences and good practices as well as to work together in solutions to solve detected anomalies.

Likewise, instructors and learners work together in learning activities, which are naturally deduced from the plan, defined in the PMP. Indeed, as the project has been validated based on the learning objectives, the learning activities to be carried out in order to construct the software project can be directly deduced from the PMP. This is particularly important to guarantee the rationalization of the learning objectives and this is another important benefit offered by the yPBL methodology. Students and instructors work together to define and plan learning activities to be carried out in every iteration of the process. In order to efficiently carry out these learning activities accordingly with the plan, both actors need to

participate in the research and preparation of the learning material. Instructors work together on the definition of a list of learning subjects. These subjects will be prepared and presented by both students and instructors. In order to facilitate the preparation of the learning material, bibliographic "Research Resources" need to be identified and proposed by both actors. In order to guarantee that the learning material to be produced is compliant with the plan, resources and project requirements, an approach based on the elaboration of "Cookbooks" can be followed. "Cookbooks" are aimed at proposing an efficient presentation of definitions and concepts (i.e. the ingredients), and how they can be applied to construct a particular software function or service (i.e. the recipes). Recommended resources and links are also proposed in the cookbook. Instructors and students carry out the preparation of the cookbooks and specific timeslots are reserved to allow them to present these learning materials. The cookbooks are also stored in a common repository in order to facilitate its access during the project process.

This is another benefit offered by the yPBL methodology. Indeed, internal actors play the roles of active and passive learners, working individually or within groups in learning activities including bibliographic research, course preparation and presentation. Moreover, the evaluation of these activities is carried out by the peers based on the real knowledge acquisition. Furthermore, during the software building process, internal actors need to apply their acquired knowledge in constructing the real software solution.

Finally, for every PBL iteration the evaluations is carried out based on the SRS, SDD and STD documents. From these documents the achievements can be objectively measured based on the requirements identification, solution design and implementation, and the test performed on the final product. Results of the evaluation in every iteration are stored in the "Course Evaluation" document.

The last activity considered in the yPBL process is the final evaluation. This final evaluation activity is carried out after the last process iteration. During this activity, all the process actors are asked to participate in a final project presentation including the final version of the project documents as well as the delivery of the final release of the project. During this activity, the functional and technical project requirements are finally measured, as well as the global satisfaction of the internal and external actors. The process itself is discussed and a list of suggestions and remarks are compiled and included in the "Course Evaluation" document. This information is very helpful to improve the process for future projects and also to measure and compare the final results.

IV. STUDY CASE

The yPBL methodology has been designed based on the experimental results obtained by applying PBL methodologies for software engineering projects in the INSA at Toulouse in France. During the last 2 years, the yPBL methodology has been experimentally used with students of 4th and 5th year of software engineering. These courses can be classified in two categories: introductory design and development courses and advanced technology oriented courses. For both categories of courses, the yPBL has been successfully followed.

In order to propose a friendly interface following the methodology, a template course has been defined using the Moodle learning management system [12]. This template is illustrated in the Figure 6.

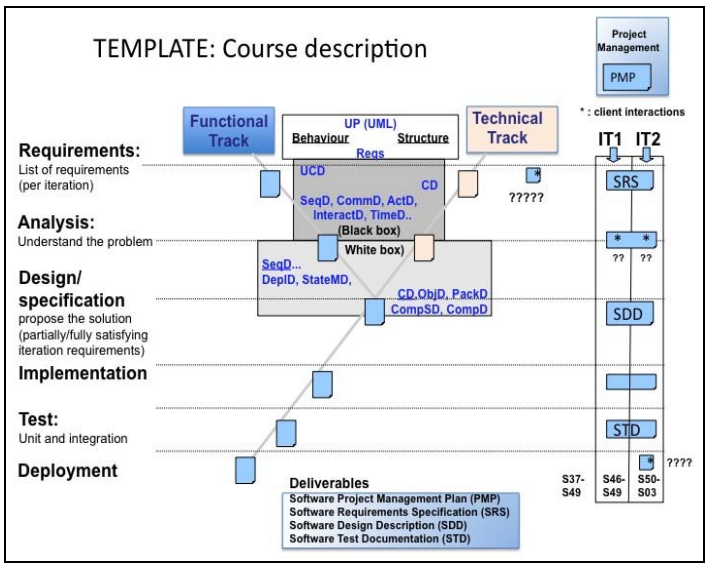

Figure 6. yBPL template for moodle interface

This template is used in order to facilitate the interaction between the various actors of the process. The interface illustrates the "y" software project methodology with the functional and technical tracks including the requirements and analysis phases as well as the merging central track aimed at designing, implementing, testing and deployments phases. For every track and iteration, explicit indications of document deliverables links are included (PMP, SRS, SDD and STD). Likewise, links to the documents produced during the learning process in the form of cookbooks are also included in the interface. This interface is configured during the start up process, and the "Course Objectives" and "Project Proposal" documents are also included. The interface is adapted to the various project and learning process actors in order to facilitate their collaboration during the whole process.

In order to illustrate a real instantiation of the methodology, a concrete study case based on a Software Oriented Architectures (SOA) systems course will be presented. Details of this study case are presented in the Table 1.

Actor	Description
Client	Direction of the DGEI-INSA department
Instructors	4 Software engineering instructors 6 Software oriented architectures instructors 3 English instructors
Students	60 students of the 5th year of IT and Networking engineering
Coordinators	1 instructor and 5 students
Learners (A/P)	5 instructors and 60 students

Table 1. Actors participating in the SOA course

The client proposing the project is represented by the GEI department of the INSA. The project proposed by the client asks for a system able to automate financial accounting activities. In the process a number important of users, interface and data needs to be considered to build the system.

The instructors participating in this course are divided in 3 groups: software engineering, SOA and English instructors. The first group of instructors targets the software engineering process and the second group targets the software technologies to be used to design and develop the software solution. English teachers participate actively in the project, working with the students in the elaboration of the documents in English versions. Moreover, English teachers supervise and guide the students in activities aimed at writing and presenting the various cookbooks developed during the learning activities and related to the project software requirements. The students participating in the project are divided in two categories: IT and Networking engineering students. Students work in groups of 12 students and each group can work in a different sub-system of the global project. The group of coordinators is composed of 1 instructor and 5 students, one student-coordinator for each groupe of 12 students. Finally, the learners groups are composed of all the students and 5 instructors mainly working in the area of software technologies.

The objective and subjective measurement results obtained from the application of the yPBL methodologies have been highly motivating. Indeed, instructors and students consider as very positive the gained experience from working in a real software project. Moreover, instructors remark the high motivation of the students, particularly when they perform learning and teaching activities. Furthermore, even if at the end of the project the full set of clients requirements have not been satisfied, students are able to explain the reasons for this partial result. They claim to have understood that living the process is the best way to know how to do things working and how to avoid in the future making the same mistakes.

CONCLUSIONS AND PERSPECTIVES

This paper has presented an innovating learning methodology, based on the well-known PBL method and inspired and adapted to software engineering unified processes. The yPBL model describing the use cases driving the methodology as well as the various internal activities guiding the process has been presented. This model has specified the relationship between the roles and phases considered in PBL methods and the roles, iterations and phases considered in the Two Tracks Unified Process (2TUP) or "y" methodology. The yPBL methodology has been defined as a process where incremental and iterative phases are performed and specific communication channels are established by the way of standard documents in the framework of a real software project. A case study illustrating how this methodology has been instantiated at the INSA of Toulouse has also been presented. Motivating results have been obtained during the experimental application of the yPBL. At the moment of writing this paper, a new instantiation of the methodology has been started and a more important set of measurements will be performed in order to better analyze and evaluate the benefits offered by yPBL.

ACKNOWLEDGMENT

The yPBL methodology has been successfully designed, implemented and experimented thanks to the valuable and active participation and collaboration of the GEI and CSH departments at the INSA of Toulouse, including direction, administration staffs as well as the teachers and the students participating in this project.

REFERENCES

[1] Savery, John R. (2006) "Overview of Problem-based Learning: Definitions and Distinctions," *Interdisciplinary Journal of Problem-based Learning* : Vol. 1: Iss. 1, Article 3.

[2] John Biggs (2003). Teaching for Quality Learning at University Buckingham: The Society for Research into Higher Education and Open University Press, ISBN 0-335-21168-2

[3] Jacobson, I.; Booch, G.; Rumbaugh, J. The Unified Software Development Process (Addison-Wesley Object Technology Series); Addison-Wesley Professional: 1999.

[4] Pascal Roques et Franck Vallée, « UML en action », Editorial Eyrolles, february 2000, ISBN-10: 2212091273

[5] PMP: Software Project Management Plan (PMP) IEEE 1058

[6] SRS: Software Requirements Specification IEEE 830

[7] SDD: Software Design Description IEEE 1016

[8] SQAP: Software Quality Assurance Plan IEEE 730

[9] SCMP: Software Configuration Management Plan IEEE 828

[10] STD: Software Test Documentation IEEE 829

[11] SVVP: Software Validation & Verification Plan IEEE 1012

[12] Moodle Learning Management System, http://moodle.org/