Transactions of the SDPS:
Journal of Integrated Design and Process Science
18 (2), 2014, 77-95
DOI 10.3233/jid-2013-0014
http://www.sdpsnet.org

yPBL: An Active, Collaborative and Project-Based Learning Methodology in the Domain of Software Engineering

Ernesto Exposito[ab*]

[a] *CNRS, LAAS, 7 avenue du Colonel Roche, F-31077 Toulouse, France*
[b] *Université de Toulouse, UPS, INSA, INP, ISAE, LAAS, F-31077 Toulouse, France*

Abstract Software engineers face the challenge of working in a very dynamic and rapidly evolving context requiring the continuous acquisition of knowledge with new software technologies, paradigms, approaches and methodologies. To address these needs, this paper proposes an active, collaborative and project-based learning methodology that is well suited to the software engineering (SE) domain. The yPBL methodology is defined as a specialization of a SE process named 2TUP or "y" that is mainly characterized by the separation of concerns between the requirements and constraints of a software product and the technologies used during its design and development. This SE process has been specialized to define project-based learning courses and the required activities allowing the research, discovery, acquisition, transmission and application of knowledge by the learners following actively and collaboratively the software product engineering process. The yPBL methodology can be applied to standard classroom courses as well as to geographically distributed courses involving students and instructors as well as professionals that need to update or acquire new SE knowledge. In this paper the yPBL methodology is formally specified using the Unified Modelling Language (UML). This methodology is illustrated and evaluated by a case study involving students and instructors of a classroom based software engineering and service-oriented computing course at the INSA Engineering School in Toulouse, France.

Keywords: Software development process, problem/project based learning, software engineering process, collaborative workspace, semantic web

1. Introduction

The Software Engineering (SE) domain involves complex software development processes demanding from analysts, designers and developers a high level of knowledge and expertise in diverse areas including project management skills, communication, design or development. Moreover, the large diversity of software design and development approaches and paradigms as well as the accelerated development of new software technologies requires continuous knowledge acquisition. This is not only the case for software engineers professionals but also for academic instructors teaching software design and development courses.

The problem based learning (PBL) methodologies have been successfully used in different domains and its benefits have been widely demonstrated (Biggs & Tang, 2011; Savery, 2006; Savery & Duffy, 1995). This learning approach asks for the active participation of the students in the learning process, playing not only the traditional role of passive and information consuming learners but also an active role through

* Corresponding author. Email: ernesto.exposito@laas.fr

 Published by IOS Press

which part of the knowledge to be acquired needs to be discovered and applied by the students themselves in the context of a given problem or project (Barron *et al.*, 1998; Blumenfeld *et al.*, 1991; Thomas, 2000),. 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.

Although PBL has been designed to be easily adapted to any educational domain, the specificities of software engineering courses need to be carefully considered in order to improve the benefits of this learning method while applying successful processes and good practices that are specific to this domain. As previously introduced, software engineering courses require the practical acquisition of a large amount of knowledge and skills. 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 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 and continuous integration techniques and finally software deployment and maintenance strategies, including the related documentation production. The previous list is not exhaustive but shows the degree of complexity involved in designing courses allowing learners to play and understand the challenges involved in the large spectrum of SE roles and facets. These courses require a well-adapted learning methodology able to cope with this complexity while integrating an efficient learning engineering process.

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 by providing an efficient process based on an incremental and iterative sequence of phases (Jacobson *et al.*, 1999). These phases include analysis and specification of requirements, design of the software solution and implementation, test, integration and deployment of the software product. 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 has been specialised in new context-specific methodologies such as the Rational Unified Process or RUP (Kroll & Kruchten, 2003), Enterprise Unified Process (Ambler *et al.*, 2005) or the Agile Unified Process (Ambler, 2002). Another interesting specialisation known as the 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 (Roques & Vallée, 2004). This process, also known as the "y" process due to its graphical representation, proposes a differentiation of two tracks for the Unified Process, the first (left) track represents the functional aspects related to the software product and the second (right) track the technical aspects (e.g. technology, environment, platforms). This separation of concerns helps software engineers to concentrate on discovering and specifying the 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 serialised activities 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 approach and adapted to software engineering processes by including a 2TUP process specialisation. This methodology, named yPBL, aims at being applied to develop software engineering courses (classic classroom or modern distributed online courses) put in the context of real software projects. The yPBL process maps the roles to be played by the participants as well as the phases, activities and tasks considered in the PBL approach into the roles and phases considered in the "y" process.

The rest of the paper is organised as follows. Section II introduces the methodology foundations related to the software engineering domain. Section III presents the UML-based yPBL methodology specification. The UML model describes the structural and behavioural aspects of the methodology including the interactions between the different actors and the information exchanged during the learning, software construction and evaluation process. Section IV presents the yPBL semantic data model guiding the active, collaborative and project-based learning activities of the methodology. Section V describes a concrete study case illustrating the use of yPBL in Software Engineering (SE) and Service Oriented Architectures (SOA) courses at the INSA engineering school at Toulouse, France. Finally, several conclusions and perspectives of this work are presented.

2. Methodology Foundations

In this section the main foundations of the yPBL methodology are introduced. The first sub-section reviews the main standard of software engineering processes represented by the Unified Process. Secondly, the description of the "y" or 2TUP process specialisation is presented. Finally, the main IEEE standards driving the process, activities and documents in the software engineering discipline are introduced.

2.1. Software engineering process

A software process defines the steps required to create a software product as well as the artefacts that are produced and consumed during the process. One of the most mature and well-known software engineering process is the Unified Software Development Process, also known as UP or USDP process (Jacobson, *et al.*, 1999). UP was introduced as a standard process for creating software products based on the use of the Unified Modelling Language (UML).

UP 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.

UP 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 UP.

There are three major characteristics differentiating UP 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 kind of software system being developed, the organisation involved, competence levels of development teams or the project size. Examples of these specialisations are Rational Unified Process (RUP), Enterprise Unified Process (EUP) or Agile Unified Process (AUP). In general, most of the processes used today for designing and developing software systems are commonly based in the principles proposed by the UP process. This is also the case for the 2TUP process described in the next section.

2.2. The "y" or 2TUP process

As previously introduced, the "y" Process or Two Tracks Unified Process (2TUP) has been proposed to face the reality of the constant change of requirements and technologies of nowadays software systems (Roques & Vallée, 2004). The "y" process is defined by two parallels tracks aimed at capturing functional and technical requirements, followed by one common centralised development track (see Fig. 1).

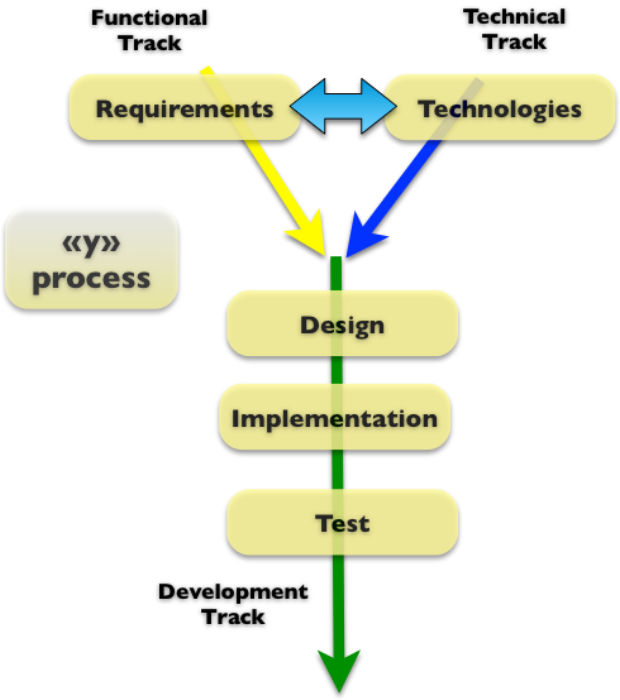

Fig. 1. "y" or 2TUP process.

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. This sequence of parallel and serialised activities can 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 facing the invariant reality of continuous changes of requirements and technologies. By separating the concerns between functional and technical requirements, engineers can apply the same process to build or maintain a software product by integrating new requirements or constraints and the most recent and well-adapted technologies.

2.3. Software engineering standards

Important efforts have been invested by the SE community to produce standards intended to drive and document software engineering processes. The most widely used in industry and academy 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 (Software Engineering Standards Committee of the IEEE Computer Society, 1998c).
- Software Requirements Specification (SRS): specifies the structure and necessary qualities of software requirements specification documents (Software Engineering Standards Committee of the IEEE Computer Society, 1998d).
- Software Design Description (SDD): proposes the necessary information content and recommendations for software design descriptions (Software Engineering Standards Committee of the IEEE Computer Society, 1998b).
- Software Quality Assurance Plan (SQAP): specifies the format and content of software quality assurance plans (Software Engineering Standards Committee of the IEEE Computer Society, 1998g).
- Software Configuration Management Plan (SCMP): describe the structure and content for a software configuration management applying to the entire life cycle of the software (Software Engineering Standards Committee of the IEEE Computer Society, 1998a).
- Software Test Documentation (STD): this document defines the form of a set of documents for use in defined stages of software testing (Software Engineering Standards Committee of the IEEE Computer Society, 1998e).
- 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 (Software Engineering Standards Committee of the IEEE Computer Society, 1998f).
- These standards help to express and communicate in a unified way all the information related to the software process.

A well-adapted software engineering learning methodology should integrate the previously introduced software processes and standards in order to guarantee efficient learning objectives achievement while actively and collaboratively building real software products. Next section introduces the yPBL methodology process model.

3. 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 used in the context of software engineering courses based on the construction of a real software product. 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 (see Fig. 2).

As an intent to formally describe the yPBL methodology that will be presented in the following paragraphs, the UML language has been used to specify the behavioural and structural views of the methodology.

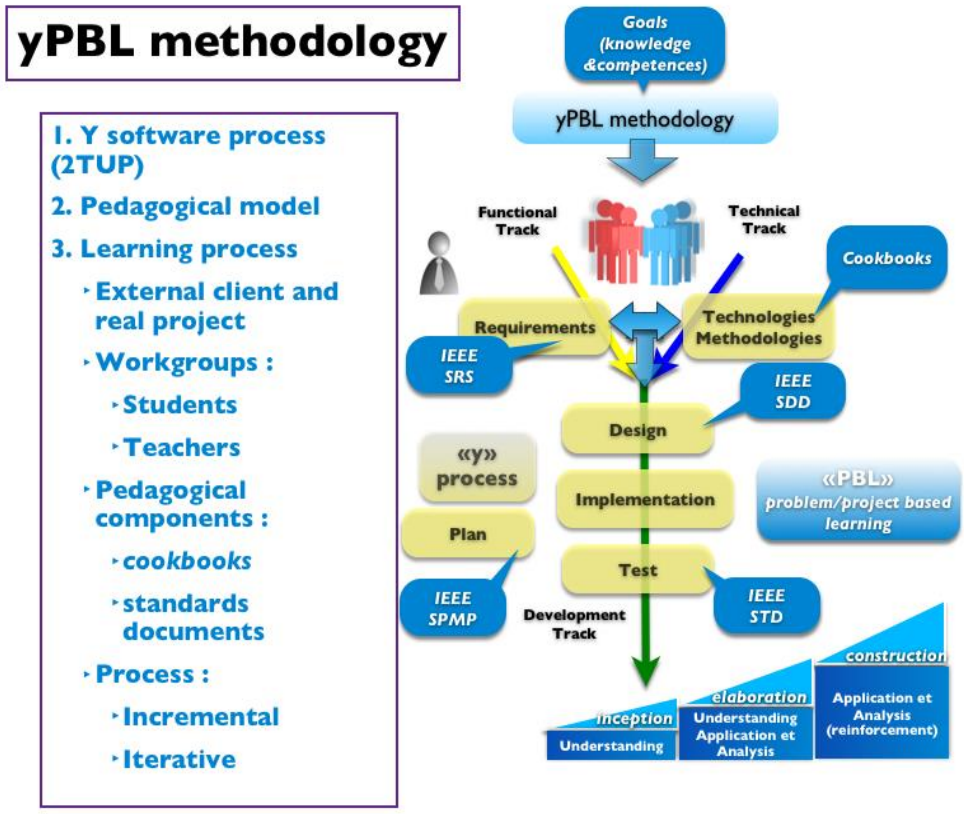

Fig. 2 Overview of the yPBL methodology.

3.1. yPBL use cases

In order to follow the software engineering good practices 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 Fig. 3.

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

Guided by the construction of the software project, two generalisations 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.

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

Generalisations of the learner role are defined by passive and active learner 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. The active role is played based on the collaborative production, refinement, dissemination and application of learning material by instructors and students during the continuous learning process.

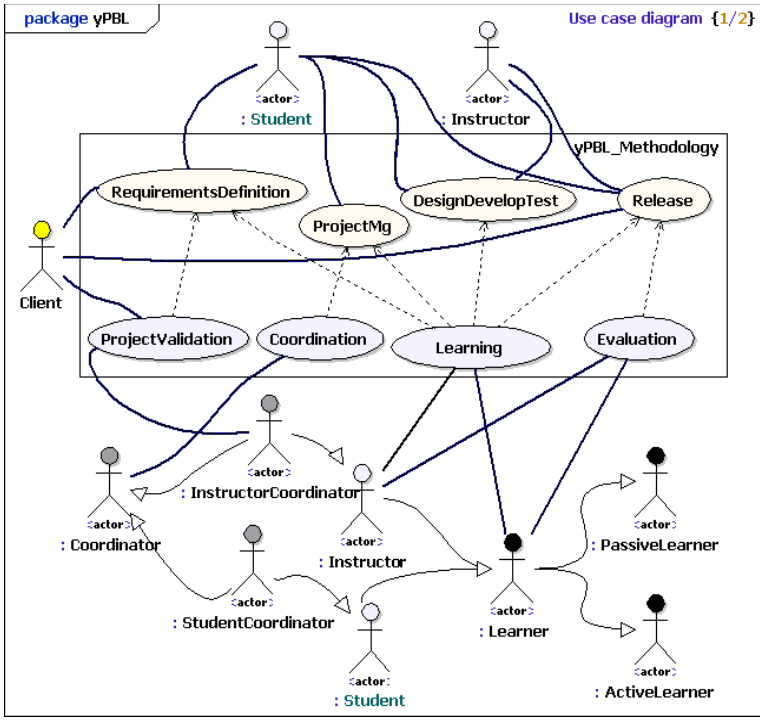

Fig. 3. yPBL methodology use cases diagram.

3.2. 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 Fig. 4.

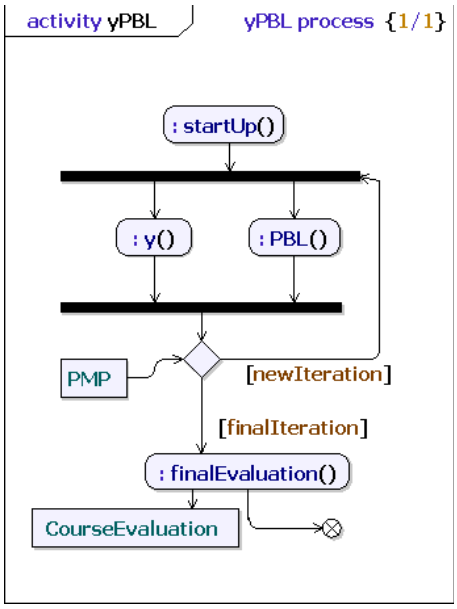

Fig. 4. Process proposed by the yPBL.

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 IEEE 1058, IEEE 730 and IEEE 828 documents, and it is intended to control and manage the yPBL process. As illustrated in Figure 4, 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. Several iterations can be defined based on the complexity of the project and the duration of the course. These iterations are mainly organised in four consecutive phases as recommended by the Rational Unified Process (Kroll & Kruchten, 2003).

- Inception: short and initial phase aimed at understanding the business cases of the system to be developed, initial overview of potential technologies to be used and to start the project management activities (e.g. planning, resources or responsibilities). During this phase the requirements have to be identified and validated.
- Elaboration: analysis of requirements and initial design of the potential solution, exploration and evaluation of the technologies to be used, preparation of validation tests, global project management activities and scheduling of software product releases. During this phase several iterations can be planned in order to research and develop basic proofs of existing technologies that could be applied to satisfy the project requirements. This work is actively and collaboratively developed, reviewed and evaluated in the yPBL methodology based on the use of a learning instrument named cookbook, which will be introduced in the next section.
- Construction: System architecture design, implementation and tests of functionalities and releases of the product. During this phase, several iterations can be planned in order to release incremental versions of the product. When following the yPBL methodology, the cookbooks elaborated during the previous phase are used to design, develop and test the product releases.
- Transition: Maintenance and evolution of the system. This phase is started when the product has been released in order to fix anomalies or to cope with new requirements or constraints. When the yPBL methodology is applied, new cookbooks can be elaborated and used to produce new product releases during the transition phase.
- Before the completion of an yPBL-based course, a final evaluation activity is carried out. During this final activity, learners and coordinators participate in a global learning project evaluation including not only the delivered software product and the satisfaction of the requirements but also the efficiency of the process and the work carried out by all the participants. Results of these evaluations are reported in the "Course Evaluation" deliverable.

3.3. 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 the sequence of activities, interaction between the various process actors, as well as communication channels are specified.

Figure 5 illustrates the start-up activity. This initial activity is performed as an interaction between the client and the instructor-coordinator.

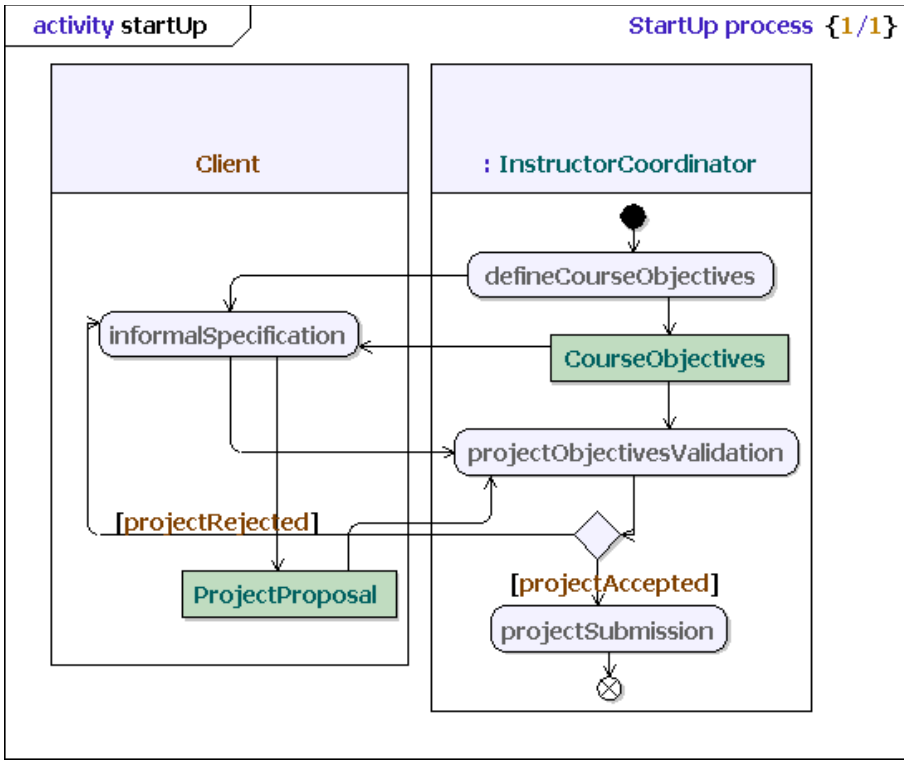

Fig. 5. Starting up process.

The start-up activity begins 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 needs to be validated 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 Fig. 4, two parallel processes guiding both software and learning project activities are started. Figure 6 illustrates the activity diagram modelling the software project process. Students and instructors perform collaborative or individual activities for every iteration of the y process. In order to stimulate autonomy, 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 (IEEE 830). Likewise, students are asked to work on the PMP document in order to define the plan to be executed within a sequence of 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.

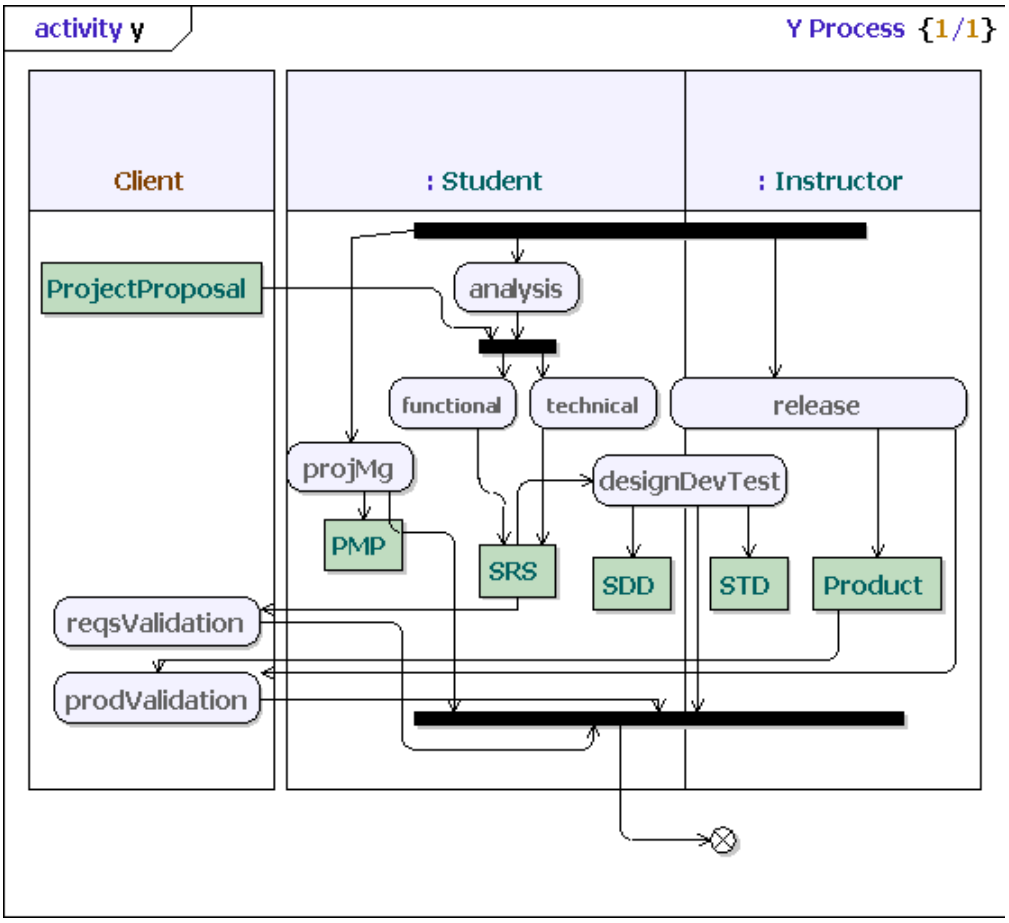

Fig. 6. 2TUP or y process.

During the design, development and testing activities, both students and instructors work together in order to produce the deliverables expected to be released in every iteration. In these activities the role of the instructor is clearly separated from the client role. Indeed, the instructor plays a supporting role (in contrast to a classical teacher/evaluator 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 (IEEE 1016) and STD (IEEE 829) standards. 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 carried out for every iteration. Figure 7 depicts the activities performed by the coordinators, instructors and learners during the PBL process.

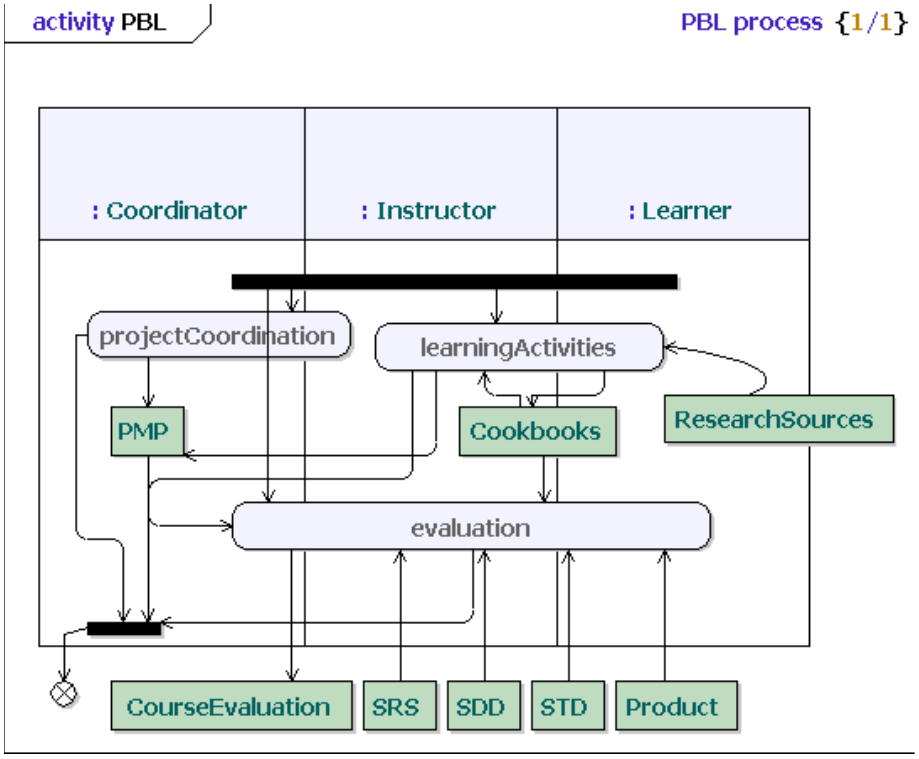

Fig. 7. PBL process.

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 minimising potential risks. Coordinators work together on the basis of periodic meetings or using any asynchronous communication mean (e.g. email or forums). During these interactions, coordinators exchange monitoring information collected during the process. This information can be used to share and encourage positive experiences and good practices as well as to work together in finding solutions to solve predicted or detected anomalies.

Likewise, instructors and learners work together in learning activities, which are naturally deduced from the plan defined in the PMP. Indeed, because the project has been validated based on the course 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 achievement 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 that are implicitly deduced from the software product requirements and that need to be carried out in every iteration of the process. In order to efficiently carry out these learning activities accordingly with the plan, both kind of actors need to participate in the research and preparation of 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 predefined learning objects named "Cookbooks" will 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 time slots are reserved to allow them to produce and present these learning materials. The cookbooks are also stored in a common

repository in order to facilitate its access during the project development process. These learning objects and its semantic data model will be further explained in the next section.

The use of these learning objects represents 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 and reinforce this acquired knowledge in constructing the software solution to be delivered at the end of the process.

Finally, for every PBL phase (i.e., inception, elaboration, construction and transition), review evaluation activities are carried out based on the presentation of the PMP, 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 tests performed on the cookbook learning objects or the final product.

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 yPBL process participants. The process itself is discussed and a list of suggestions and remarks are collected and included in the "Course Evaluation" document. This information is very helpful to improve the process for future projects (i.e. best practices) and also to measure and compare the final results.

4. yPBL Semantic Data Model

In this section the yPBL semantic data model to be used for the collaborative knowledge searching, production, dissemination and application will be introduced. In the framework of problem-based or project-based courses, students generally work in groups on problems or projects which are close to professional life: the situation must be realistic, credible and engaging for them. From the situation-problem, the students must reflect on what precisely is entailed, what they know, and what they need to learn and do to be able to provide a solution (Hmelo-Silver, 2004). The students' progress through the project alternating between individual work sessions and group sessions when they come together to pool what they have learnt and move to the next step in finding the solution. Mechanisms are integrated whereby individual students and the group can assess their progress in their learning and in completing the project.

As previous described (see Fig. 3), the main actors involved in yPBL courses are the instructors, responsible for the course material provided to the participants, the students, the coordinators whose role is to guide the activities carried out by the learners, and the 'client' who intervenes periodically to clarify project requirements and constraints. The data model of a well-designed yPBL course plays a crucial role, in particular to guarantee an efficient collaboration driven by the project requirements (and implicitly by the learning objectives) and allowing the dynamic collaborative knowledge production and consumption between all the participants.

During the yPBL course, the participants are grouped within teams working on the real software project proposed by the external "client". Each team is composed of learners, instructors and coordinators working together and playing different roles during the process (project coordinator, software architect, software developer, software tester, quality manager). The first task to be accomplished is to understand and if necessary reformulate the specifications of the software that needs to be developed. Then, as a team, they need to explore the different possible technical technologies that exist, learn how to use them and then select the ones they will use when actually developing the software product.

As previously introduced, in order to facilitate these stages, yPBL uses a pedagogical object called "cookbook". Cookbooks are aimed at proposing an efficient collection of definitions and ingredients and how they can be applied to construct specific software components, functions or services (i.e. the recipes) that can be followed to satisfy the project requirements. Initial "root" cookbooks can be produced by the instructors in order to give high-level overview of the involved technologies, methodologies, best-practices

and project-specific solutions and strategies. From these initial cookbooks, the project specific cookbooks are produced in complete autonomy by students (in pairs) from documentary research based on the functional and technical requirements of the project. Each cookbook is pre-published in the yPBL collaborative workspace in order to be peer-reviewed and modified correspondingly before being published to all the course participants in the yPBL cookbook repository. In this way, the cookbooks will be available to all the participants during the design and development phases of the project. In the framework of yPBL courses, public conferences are usually organised during the elaboration phase in order to allow the students to present and demonstrate their cookbooks to the whole project team.

Throughout the project, yPBL places students in situations where they must assume an active role in their learning when they need to acquire knowledge and skills themselves and subsequently apply what they have learnt. With the cookbooks and conferences, peer instruction/learning is formalised. The students are required 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. Figure 8 presents the yPBL cookbook data model. All the cookbooks are organised by domains, including methodological (i.e., software engineering, service-oriented architectures, object-oriented design, etc.) or technological (i.e., mobile, multimedia, distributed, enterprise applications, etc.) domains.

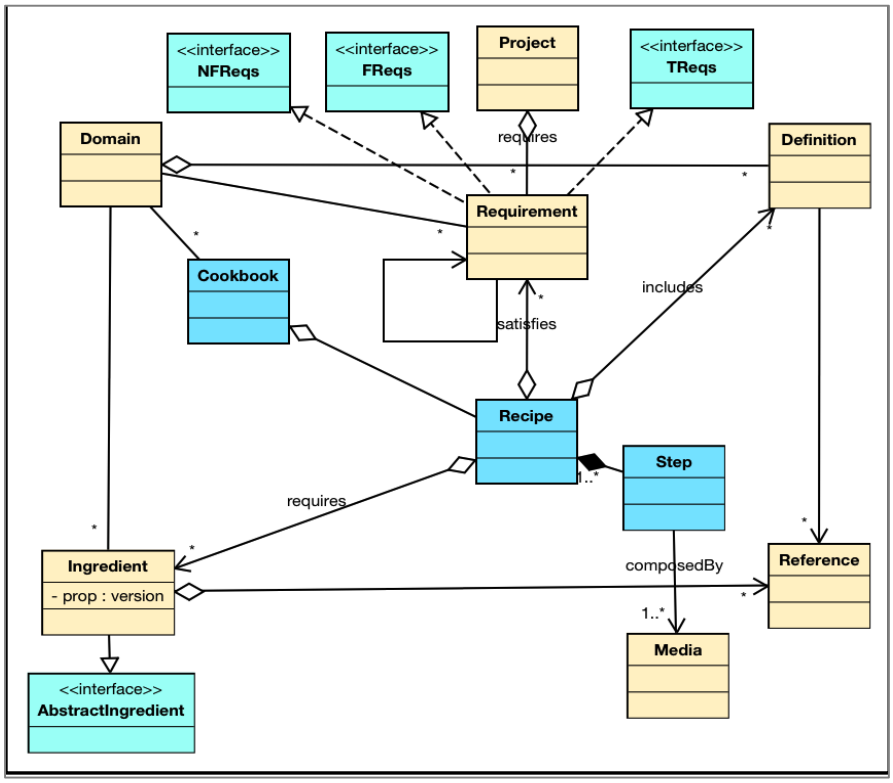

Fig. 8. yPBL cookbook data model.

For each cookbook a set of recipes will be developed in order to demonstrate the feasibility of satisfying functional and technical requirements, including specific ingredients and involving domain specific definitions. Each recipe is developed as a set of steps that are described using multimedia elements including text, pictures, videos, audio, etc. Recipes are developed as proofs of potential solutions and after being reviewed and presented to the whole team, one subset of recipes will be selected and applied by all the participants in order to implement the final software product.

The yPBL cookbooks repositories used to provide the adequate collaborative workspace have to implement this data model. Cookbooks repository implementations need to integrate an adequate technology able to facilitate the collaboration and to guarantee that all the consumers and providers of the cookbooks share the same semantic of definitions, ingredients and requirements. yPBL cookbooks repositories can be designed to use technologies such as semantic wikis in order to provide a flexible way to produce and consume content that can be annotated based on the yPBL data model. The open source Semantic MediaWiki (SMW) extension has been efficiently evaluated to implement yPBL collaborative workspaces. SMW is a free open source extension of Mediawiki (the wiki engine of Wikipedia) that is well-suited for collaborative workspaces including the advantages of semantic web technologies. The content collaboratively produced within SMW can be encoded using standard RDF (Klyne *et al.*, 2004) and OWL (McGuinness & Van Harmelen, 2004) languages. In such way, repository content can be consumed using semantic browsing and queries features facilitating the research and the discovery of solutions by participants that share the same semantic model of functional and technical requirements. In such way, each yPBL-based course can take advantage of common cookbooks repositories that have been elaborated by other teams working on other projects at the same time or during previous courses. The process of efficiently exploring and using the cookbooks repositories can be considered as being part of the learning acquisition and evaluation process, as the learners need to have a good understanding of the knowledge model that has been semantically annotated within the cookbooks and recipes. Based on this exploration, they can decide what to use, to adapt or to extend in order to implement the required solution. During the cookbooks elaboration phase, the learners and coordinators will decide how to adapt existing recipes or to create new recipes when the project requirements cannot be satisfied by the recipes that have been produced for other projects. In this way, an even larger collaboration with previous, concurrent or future project-based learning courses can be guaranteed by yPBL cookbook semantic-driven repositories.

5. Case Study

The yPBL methodology has been iteratively and incrementally designed based on the experimental results obtained by applying PBL methodologies for software engineering projects in the INSA Toulouse in France. During the last 5 years (2008-2012), the yPBL methodology has been experimentally applied to software engineering and service oriented architectures courses.

In order to propose a dashboard interface following the methodology, an yPBL template course has been defined to be used in the Moodle learning management system (Moodle, n. d.). This template is illustrated in the Fig. 9.

This template is intended to facilitate the interaction between the various actors within the several phases of the yPBL methodology. In the general section, information such as course goals, planning, participants, links to the resources repositories as well as synchronous (e.g., audio/video conferencing or chat components) and asynchronous (e.g., forum or messaging components) communication channels are presented. The yPBL methodology section introduces various resources describing the yPBL methodology, its process and document templates. The project section is intended to publish the specification of the project and its requirements, as well as to present communication channels with the client. Finally, inception, elaboration and construction phases will be incrementally enabled during the project progression and will allow all the participants to produce and consume the various learning and software product deliverables.

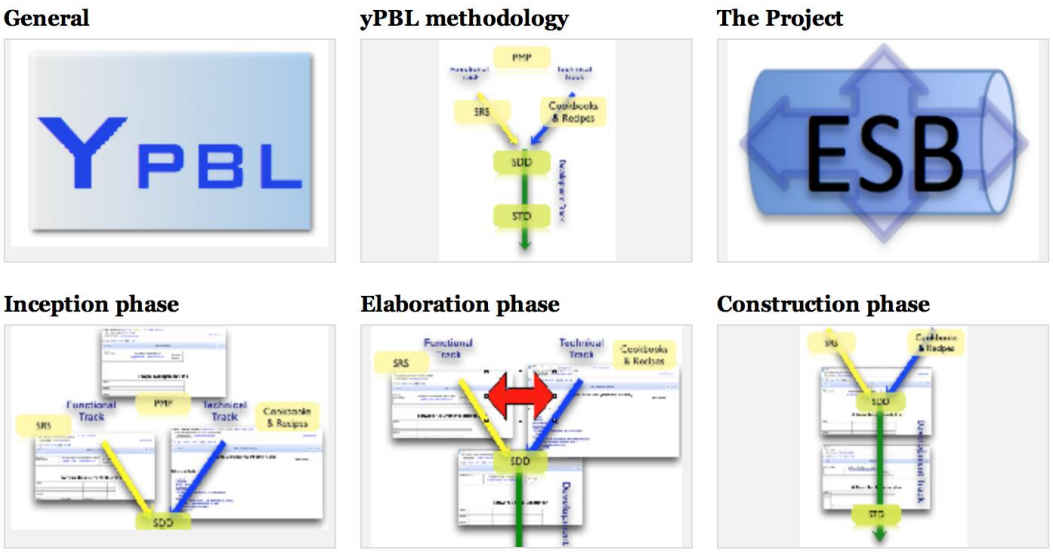

Fig. 9. yBPL template for Moodle interface.

In order to measure the benefits of the yPBL methodology, a concrete study case based on a Software Engineering (SE) and Service Oriented Architectures (SOA) courses is presented in the following paragraphs. The estimation of actors participating each year within these courses is presented in the Table 1 (i.e. approximately 11 instructors and 72 students).

Table 1. Yearly estimation of actors participating in the SE&SOA course

Actor	Description
Client	IT Services company or academic (open source projects)
Instructors	4 Software engineering instructors
	4 Software oriented architectures instructors
	3 English instructors
Students	72 students of the 5th year of IT and networking engineering (working in teams of 9-12 students)
Coordinators	1 instructor and 5-6 students
Learners	11 instructors and 72 students

During the last 5 years (2008-2012), the actors playing the role of clients and proposing the yPBL projects have been represented by local industrials or academics. The projects proposed by the clients have been generally intended to design and develop Web-based applications based on the composition and orchestration of distributed web services.

The instructors participating in these courses are divided in 3 groups: software engineering, service-oriented architecture 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 develop the software solution. English instructors participate actively in the project, working with the students in the elaboration of the documents in English versions. Moreover, English instructors 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 teams of 9-12 students and each team can work on a complete system development or on one specific sub-system of the global project.

The group of coordinators is composed of 1 instructor and 5 or 6 students, one student-coordinator for each team.

Finally, the learners groups are composed of all the students and the instructors. Indeed, instructors and students collaboratively participate in the learning process, based on the requirements that need to be satisfied by the project to be delivered to the external client. Each year, the project changes and usually new challenges need to be targeted, in particular when new technologies or specific project contexts are demanded by the clients.

Figure 10 illustrates the analysis of various surveys carried with the yPBL course participants. In this study, the survey has asked the participants to evaluate the amount of knowledge and skills acquired during the active and collaborative activities (cookbook production, reviewing and application to the project). The subjective appreciation of the participants regarding the quality of the acquired knowledge and skills is also analysed (e.g. superficial or deep/long lasting acquisition).

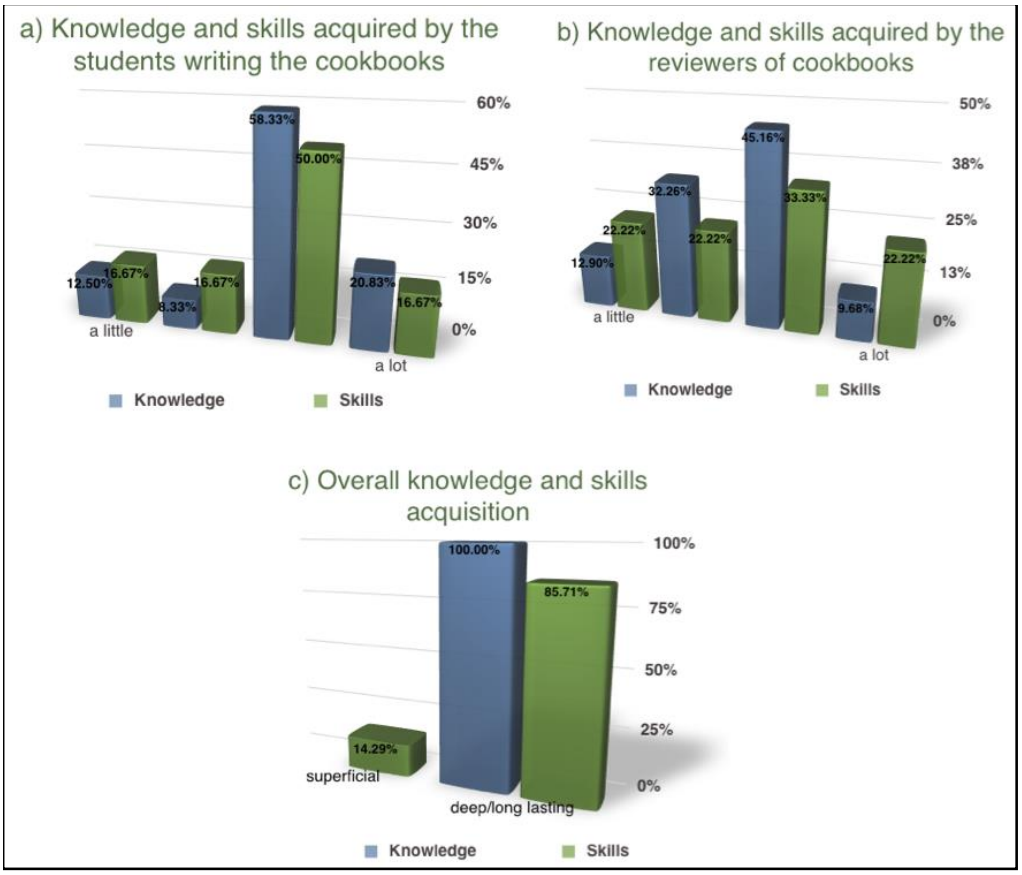

Fig. 10. yPBL participants survey results.

- Figure 10.a) illustrates the learners' point of view about the high level of knowledge and skills acquired during the collaborative process of cookbooks production.
- Figure 10.b) shows the learners point of view about the average level of knowledge and skills acquired during the cookbooks review process.
- Figure 10.c) shows the high level results about deep and long-lasting quality acquisition of knowledge and skills during the yPBL course.

The encouraging results obtained from the application of the yPBL methodologies can be considered as highly satisfactory and well adapted to the active and dynamic software engineering domain and the new generation of learners. 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 work collaboratively performing learning and teaching activities. Furthermore, even if at the end of the project the full set of clients requirements are not always satisfied, students are able to identify, explain and propose solutions to cope with the encountered difficulties. 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. Moreover, the use of modern collaborative workspaces motivates and accelerates the learning process and allows to easily capitalise knowledge from one learner to another, from one year to the next one.

Further studies will be carried out in order to analyse how the population that has participated in yPBL courses evaluate the benefits of this methodology in developing their industry, research and academic careers.

6. Conclusions and Perspectives

This paper has presented an innovative learning methodology, based on the well-known PBL approach 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 means of standard documents and learning objects in the framework of a real software project. Likewise, the yPBL cookbook semantic data model intended to facilitate the dynamic collaboration among project-based learning course participants has been presented. This semantic data model is intended to guide the consumption and production of learning objects named cookbooks and based on the rational identification of functional and technical requirements of the project. This yPBL cookbook semantic repository is not only intended to be used during the course period, but also after, when experienced or expert students can keep collaborating with new courses and can also keep learning new technologies or methodologies.

A case study illustrating how this methodology has been instantiated at INSA Toulouse has also been presented. Motivating results have been obtained during the experimental application of 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 analyse and evaluate the benefits offered by yPBL. Furthermore, new studies will be carried out in order to evaluate the yPBL methodology based on the point of view of engineers and the professional development of their industry, research and academic careers.

Acknowledgments

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 Toulouse, including clients, instructors and students participating in yPBL projects. A special acknowledgment is due to Anne Hernandez and her English teachers' team for their valuable support and encouragement throughout this project.

References

Ambler, S. (2002). *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*: Wiley. com.

Ambler, S., Nalbone, J., & Vizdos, M. (2005). *Enterprise Unified Process, The: Extending the Rational Unified Process*: Prentice Hall Press.

Barron, B. J. S., Schwartz, D. L., Vye, N. J., Moore, A., Petrosino, A., Zech, L., & Bransford, J. D. (1998). Doing with Understanding: Lessons from Research on Problem-and Project-Based Learning. *Journal of the Learning Sciences, 7*(3-4), 271-311.

Biggs, J.& Tang, C. (2011). *Teaching for Quality Learning at University*: McGraw-Hill International.

Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., & Palincsar, A. (1991). Motivating Project-Based Learning: Sustaining the Doing, Supporting the Learning. *Educational psychologist, 26*(3-4), 369-398.

Hmelo-Silver, C. E. (2004). Problem-Based Learning: What and How Do Students Learn? *Educational Psychology Review, 16*(3), 235-266.

Jacobson, I., Booch, G., & Rumbaugh, J. E. (1999). *The Unified Software Development Process-the Complete Guide to the Unified Process from the Original Designers*: Addison-Wesley.

Klyne, G., Carroll, J. J., & McBride, B. (2004). Resource Description Framework (Rdf): Concepts and Abstract Syntax. *W3C recommendation, 10*.

Kroll, P.& Kruchten, P. (2003). *The Rational Unified Process Made Easy: A Practitioner's Guide to the Rup*: Addison-Wesley Professional.

McGuinness, D. L.& Van Harmelen, F. (2004). Owl Web Ontology Language Overview. *W3C recommendation, 10*(2004-03), 10.

Moodle. (n. d.). Moodle Learning Management System, August 09, 2013, from https://moodle.org

Roques, P.& Vallée, F. (2004). *Uml 2 En Action: De L'analyse Des Besoins À La Conception J2ee* (Vol. 3): Eyrolles.

Savery, J. R. (2006). Overview of Problem-Based Learning: Definitions and Distinctions. *Interdisciplinary Journal of Problem-based Learning, 1*(1).

Savery, J. R.& Duffy, T. M. (1995). Problem Based Learning: An Instructional Model and Its Constructivist Framework. *Educational technology, 35*(5), 31-38.

Software Engineering Standards Committee of the IEEE Computer Society. (1998a). Ieee Recommended Practice for Software Configuration Management Plans (Vol. 828): IEEE.

Software Engineering Standards Committee of the IEEE Computer Society. (1998b). Ieee Recommended Practice for Software Design Descriptions (Vol. 1016): IEEE.

Software Engineering Standards Committee of the IEEE Computer Society. (1998c). Ieee Recommended Practice for Software Project Management Plan (Vol. 1058): IEEE.

Software Engineering Standards Committee of the IEEE Computer Society. (1998d). Ieee Recommended Practice for Software Requirements Specifications (Vol. 830): IEEE.

Software Engineering Standards Committee of the IEEE Computer Society. (1998e). Ieee Recommended Practice for Software Test Documentation (Vol. 829): IEEE.

Software Engineering Standards Committee of the IEEE Computer Society. (1998f). Ieee Recommended Practice for Software Verification and Validation Plan (Vol. 1012): IEEE.

Software Engineering Standards Committee of the IEEE Computer Society. (1998g). Ieee Standards for Software Quality Assurance Plans (Vol. 730): IEEE.

Thomas, J. W. (2000). A Review of Research on Project-Based Learning.

Author Biography

Ernesto Exposito is an Associate Professor at the INSA of Toulouse and he is researcher in the SARA team (Services et Architectures pour les Réseaux Avancés) at LAAS-CNRS, France. In 2004, he worked as Researcher in the National ICT Australia Limited (NICTA) research center in Sydney, Australia. In 2003, he earned his PhD in computer science and networking from the Institut National Polytechnique de Toulouse. In 2010, he earned his HDR post-doctoral degree (accreditation to supervise research) from the Institut National Polytechnique de Toulouse.

His research interests include autonomic communication services aimed at satisfying the requirements of new generation distributed applications in heterogeneous networked environments.

In the pedagogic domain, his interests includes active and collaborative learning approaches in software engineering.

He is author of more than 80 publications including international journals, regular and invited international conference papers, books and book chapters.

His home page is http://homepages.laas.fr/eexposit