An Experience of Educational Innovation for the Collaborative Learning in Software Engineering

doi:10.3991/ijet.v6i2.1662

Juan C. Yelmo and Juan Fernández-Corugedo
Universidad Politécnica de Madrid, Madrid, Spain

***Abstract*—This paper describes the experience in collaborative learning based educational innovation for the Software Engineering Subject at the Telecommunications Engineering School of the Universidad Politécnica de Madrid. We describe the context of evolution of the Software Engineering related subjects in the Master's degree in Telecommunications Engineering (specialization in Telematics) for their adaptation to the educational and organizational postulates of the European Space for Higher Education. In this context, we describe a pilot project for the adaptation of the Software Engineering Lab course, which includes the development and validation of a software engineering tool environment and a web platform for collaborative work.**

***Index Terms*—Collaborative Learning, Education Innovation, European Space for Higher Education, Software Engineering.**

I. INTRODUCTION

This paper describes the experience in educational innovation around the collaborative learning of Software Engineering related subjects at the Master's degree in Telecommunications Engineering (a five years academic degree), specialization in Telematics (computer networks, computer science and telecom services) at the Universidad Politécnica de Madrid (UPM), Spain. The current Master's degree program is now evolving in the context of the Bologna Declaration [1] convergence process into a new Bachelor plus Master degree organization.

In this context, we describe a pilot project for the adaptation of the Software Engineering Lab subject according to the educational and organizational postulates of the European Space for Higher Education.

The Software Engineering Lab is a fifth year course that follows an educational methodology based on collaborative work, continuous evaluation, close monitoring of students teams and instructional feedback. Students team up in groups of three to five members where each member is assigned a role and a shared responsibility in achieving common goals. Each team is assigned a software development case study, in general a distributed software application or telecom service, which must be carried out along a semester with the tutorship of a supervisor, the support of a software engineering tool environment and a web platform for project management and collaborative work.

The Lab follows a continuous evaluation systems based on the evaluation and assessment of technical deliverables, prototype demonstrations, oral presentations, work-plan accomplishment, peer evaluation, etc. This evaluation is made at team level, in order to foster team building and team performance, but individual students are observed to check student progress and learning gaps. The results of the evaluation process are used to monitor the learning process itself and to provide feedback to students. Finally, the students also evaluate the learning process and Lab organization and resources through questionnaires conducted at University level.

The general process of evaluation and improvement of the learning process and Lab organization and resources was intensified in the course 2005-06, in which this subject was declared a pilot experience by the general programme of the UPM for the implantation of new educational methodologies within the European Space for Higher Education and the European Credit Transfer System (ECTS).

The pilot project for the adaptation of the Software Engineering Lab course included revision of goals, contents, organisation, methodology, grading policy, effort measure for students and teachers and lab environment and tools supporting collaborative work. In this paper we describe in detail the APIS application, a web-based tool for Software Engineering Project Management developed at the Department of Telematics to support iterative software processes based on the Unified Software Development Process Model (RUP) [2]. This tool was initially designed as a front-end for the BSCW [3] collaborative environment supporting a lightweight version of the RUP. APIS has been redesigned as a standalone web application supporting remote collaboration in the definition and management of educational case studies in software development. The focus of the new design was put on better educational support, usability, robustness and easy maintenance and evolution.

The rest of this paper provides information on the academic context of the software engineering discipline within the Telecommunications Engineering degree at UPM and then goes into details about the organization and educational methodology of the Software Engineering Lab course and the goals and results of this pilot experience in educational innovation. The paper finally focuses on the tool environment supporting the learning process and in particular APIS, as an example of application supporting the collaborative learning in software engineering.

II. SOFTWARE ENGINEERING IN TELEMATICS

The Telecommunications Engineering degree at UPM (actually, core modules, structure and number of semesters for this degree are common to all universities in Spain) is a five-year university programme where students are given the option to choose specialization at the second semester of the fourth year. The three available specializations are *Electronics*, *Communications* and *Telematics* (Fig. 1).

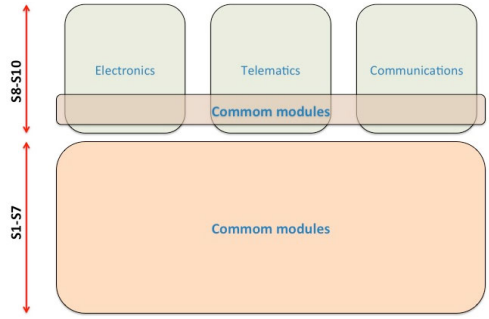

Figure 1. Structure of the degree program in Telecommunications Engineering

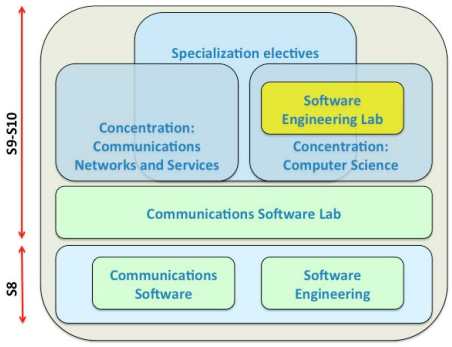

Figure 2. Structure of the specialisation in Telematics

In the case of *Telematics*, the first specific modules are *Communications Software* and *Software Engineering*. In the fifth year, Telematics student take as a mandatory module the *Communications Software Lab* and can choose between two focused concentrations: *Communications Networks and Services* and *Computer Science*. In practice, both sub-disciplines are partly open and share several elective modules. One of the common modules is the *Software Engineering Lab*, which is a mandatory module for *Computer Science* students but elective for the rest of students. Fig. 2 summarises graphically the structure of the specialization in Telematics.

In summary, most of the learning process related to the software engineering discipline takes place in the *Software Engineering* and *Software Engineering Lab* modules, within the specialisation in Telematics.

The Software Engineering module is an introductory course providing a conceptual background in software engineering that can be applied to any kind of computer-based systems. The purpose of the course is providing the students with the knowledge and abilities needed to understand and tackle the technical aspects of the discipline as well as its economic, organizational and social facets.

The three main topics of this module are:

- **The Software Development Process**. A general overview of software life cycle models and the different processes, techniques and tools involved in the development of software systems, from the requirements elicitation to the software quality assurance. The focus is put in processes and key areas as reflected in ISO 12207 [4] and SWEBOK [5].
- **Object Orientation Techniques**. A general methodology for the analysis and design of software systems based on the object oriented approach, the Unified Process Model and the Unified Modelling Language [1].
- **IT Project Management**. An overview of software project management concepts and techniques (estimation, planning, risk management, etc.) as well as economic aspects and human factors related to teamwork in software development projects [6].

The syllabus for this module is completed with a number of invited talks on advanced topics in software engineering and professional aspects given by renowned practitioners with responsibility in industrial-strength software development projects. As an example, some of the topics treated in recent years have been: The ITIL[1] standard for IT service management, core banking platform architectures and innovation management and intellectual property rights in software development projects.

The Software Engineering module has evolved in the last years towards a learning process model based on continuous evaluation and the combination of classroom lectures, guided lab assignments and group work.

Regarding the Software Engineering Lab, its main goal is to reinforce the learning process in software engineering by providing skills and practical abilities related to techniques, methodologies, industry standards, tools and notations widely used in the industrial development of software systems.

In this module the application field gets more concrete, focusing on telecommunication systems and information society services.

Next section provides more details on the organization and learning process of the Software Engineering Lab.

III. THE SOFTWARE ENGINEERING LAB

The Software Engineering Lab is a hands-on, project-oriented course in which students tem up and freely organise to work in a case study either on-site at the Lab or remotely through Internet. Case studies are intended to be realistic and motivating information society services, which must be analysed, and designed by the Lab teams. This includes a proof-of-concept prototype and a final demonstration. Nevertheless, the focus is not put on the implementation of a product or service but rather on the development methodology, thus limiting the implementation effort by providing the student with pre-existing service components and telecom service emulators. Case Studies must be developed using processes, tools and techniques in common use in the software industry. In particular, the Lab assignments are intended to emphasize the learning of the following software engineering topics:

- Management and planning of software development projects
- Collaborative coordination and team building
- Software configuration management
- Use of standards for modelling and documentation of software systems
- Architectural design of telecom services
- Object-oriented design of software systems
- Use of tools supporting the different phases of the software development process

[1] http://www.itil-officialsite.com

Students' teams are formed by a group of four or five members and share work and responsibility by assigning each member a main role. Name and responsibilities for each role depend on the process model chosen for the development of the case study, but in general team members play one of the following main roles:

- **Project manager**. Coordinates the team and has responsibility on project planning and monitoring. The project manager is also responsible for representing the team, negotiating restrictions and presenting results.
- **Integrator**. He/she is responsible for the configuration management and integration of software modules and components.
- **Analyst**. Elicits and interprets system requirements, formalize these requirements into analysis models and elaborates the system specification.
- **Designer**. He/she is responsible for the elaboration of a system design for the case study and develops a plan for a solution in terms of software components. Usually designers are also responsible for system implementation.

Once the team is assigned a case study, they can proceed with the project by using the process model and the set of techniques, notations, tools and standards proposed by the supervisor.

Regarding the process model, we have used several life cycle models in the past: waterfall model, unified model, lightweight agile models, etc. but from the pilot project we decided to adopt a lightweight iterative model tailored from the Unified Process Model.

The students present their results and achievements along the semester by a number of deliverables and public presentations. These deliverables consist of technical and management documents, analysis and design models, source code, test cases and in general any intermediate or final software artifacts that represent a relevant outcome of the project. The time schedule for milestones and the list of deliverables depends upon the concrete process model and the project plan elaborated by the team at the beginning of the semester.

Lab teams give one or two intermediate presentations for supervisors and the rest of students in order to report about project progress, justify design decisions and selection of tools and technologies and also to try to convince the audience about the advantages and relevance of the system under development and the adequacy of the implementation approach. There is also a final presentation at the end of the semester, which also includes a demonstration of the system or service developed. It is in general a limited operational prototype implementing the main uses cases of the system.

The Software Engineering Lab uses a continuous evaluation approach for the learning process based on submitted deliverables and presentations and the corresponding instructional feedback. Feedback is important to acknowledge students for their effort, provide reinforcement for their accomplishments and explain the students what they are doing that is correct or incorrect.

Students also participate in the evaluation process by providing evaluation and feedback after the final presentation of the case studies (but for their own case study). For this purpose they fulfil an evaluation form that provides instructions on how to assess different aspects of the presentation: goals, organisation, contents, communications skills, use of multimedia resources, rhythm and duration, interaction with audience, demonstration, etc.

IV. TOOL ENVIRONMENT SUPPORTING THE LEARNING PROCESS

The Department of Telematics Engineering at UPM has two lab areas that all together provide about 160 computers that are available for lab classes and work assignments. There are computers with different operating systems including Windows, Linux and Mac OS X and all of them can be managed and restarted remotely, even reloading a clean operating system image from the network after failure o malicious manipulation. The operating system image also includes a set of common tools for all the courses of the department: office applications suite, e-mail, web browser, etc.

The lab network has several general-purpose servers supporting the learning process and the management and operation of the lab itself: system administration, Web server, Moodle server, access control, binaries server, routers, etc. The access control server provides authenticated remote access of students to the lab resources from anywhere on the Internet (Web and Virtual Network Computing). Actually, the remote access is the most common way to access lab resources nowadays, since most of the students have their own computer and wide-band Internet access, allowing them to work on lab assignments at home while collaborating with other classmates participating in the same assignment.

Besides these common lab resources, each course in the department can have its own virtualised server with specific tools, servers, execution environment, users management policy, etc. This is the case of the Software Engineering Lab, which has its own server with specific tools environment for the development and deployment of the case studies and the specific collaboration tools supporting the learning process. The specific development tools in the server change quite frequently depending on the technology evolution and the specific case studies proposed each academic year. As an example, some of the tools and components installed in the last term were the following:

- Development and execution environment: Java runtime environment, Eclipse with specific plug-ins, Subversion, Apache Tomcat, Axis, MySQL, PHP5, etc.
- Telecom service components and emulators: SMS-Center, MMS-Center, Position service (Ericsson MPS), Web desktop for cloud computing services (eyeOS), etc.
- Development toolkits for mobile terminals: Java ME, Android, etc.

Educational licenses of commercial software engineering tools are also available, such as IBM RSA (UML based development) and IBM Doors (Tool supporting the requirements engineering process).

Finally, the Lab has specific information and collaboration tools supporting the learning process: A Web site with public information about the course and restricted information and documentation for students and a Moodle community as Learning Management System and remote collaboration environment.

As a result of the educational innovation pilot project we have completed the tool environment with a new tool developed at the Department. It is a web-based application supporting the collaborative learning of the software engineering discipline. This tool is described in detail in the next section.

V. APIS, A TOOL SUPPORTING COLLABORATIVE LEARNING IN SOFTWARE ENGINEERING

The Unified Process model is a software development process model proposed initially by Rational Software (company acquired by IBM in 2003). This is the same company and team that elaborated the Unified Modeling Language, a standardized modeling language for software systems. The RUP is a risk-driven, iterative model that proposes to organize projects in a number of mini-projects (iterations) that incrementally complete and refine the system under development until the final version is ready for operation. Each of the mini-projects can be considered on its own as a sequential development process, in which progress is seen as flowing steadily downwards (i.e. a waterfall life cycle model). The complete project is a collection of mini-projects organized in the following four phases:

- **Inception**: Business case, project feasibility, initial project planning.
- **Elaboration**: Iterative requirement capture, implementation of main use cases, system architecture
- **Construction**: Incremental implementation of the rest of the system
- **Transition**: System deployment in the operation environment, system validation, product release.

After the thorough review of the Software Engineering Lab course in the context of the pilot project of educational innovation and Bologna convergence, we decided to adopt a tailored, lightweight version of the Unified Process Model for the Lab assignments. By lightweight we mean that only the most relevant project artifacts for the system modeling and documentation are elaborated and submitted by project teams and, on the other hand, that students only perform project activities in the Inception and Elaboration phases with the following objectives: getting a global view of system goals and requirements, establishing technical and economic feasibility, elaborating a use case based requirement capture and developing only a prototype implementing the main system use cases. The only additional learning goals are related to project management, software configuration management and tool environment activities. This is consistent with the educational nature of the Lab assignments, since case studies are not intended for commercial release or real exploitation. In this context, the milestone that, in the case of a real project, corresponds to the end of the Elaboration phase happens to coincide with the end of the academic project for the Lab teams and the final presentation of project results. Usually students choose to carry out one or two iterations in the Inception phase and two or three iterations in the Elaboration phase. This implies between three and five milestones and the corresponding deliverable submissions in a semester.

The Web-based collaborative tool developed in the Department to manage RUP-based software development process models is called APIS. APIS allows the definition of projects, participants, roles, iterations, artifacts and deliverables. APIS can be used by supervisors and Lab teams to adapt the general-purpose RUP to the specific needs of a particular case study, team organization and workplan. Along the duration of the project, APIS can be used to upload artifacts and internal documents and submit deliverables, i.e. a set of artifacts that constitutes a submission commitment at a project milestone.

APIS also allows the remote collaborative work of team members and the project monitoring by the supervisors. In its initial implementation, APIS was designed as a front-end of BSCW [3], a well known platform supporting collaborative work supporting document management for deliverables, instant messaging, calendar, notes, events, etc. then APIS was successfully used by Lab student during several courses and, in the academic year 2009-10, has been redesigned in order to improve functionality and performance and make it a stand-alone application that can be used on its own, integrated with BSCW or with any other collaborative work platform, in particular Moodle [7].

Next subsections provide a more detailed description of APIS functionality, architecture and implementation technologies.

A. Functional requirements

The main goal of the APIS application is providing the users support in performing all the tasks and management activities in a software development project according to a particular software development methodology.

The specific support functionalities depend on the type of user and the role that the user is playing in the context of the project. Currently, APIS considers three types of users:

- **Student**. This type of user includes all the students registered for he Lab course. There are two access profiles for the students in APIS: *Lab Team*, and *User*. These users will use APIS to carry out all project activities of the case study assigned to the Lab team. The roles that user in a team can play in APIS are those described above in section III.
- **Professor**. This type of user includes Lab professors and, in general, academic staff. This kind of users can access the application using two profiles: *Supervisor* and *Administrator*. These users will access APIS in order to monitor project progress and team performance and also to carry out system administration tasks.
- **Authorized personnel**. This group includes all users authorized by lab professors to perform system administration tasks.

Fig. 3 gives an intuitive overview of the functional requirements for APIS in the form of a UML Use Case Diagram. This diagram depicts four actors that have been identified from the user profiles for the different types of users described above. The characteristics of the different actors can be summarized as follows:

- **Team**. This actor represents the group of students working on a case study. Through this actor, the students in a team can create and configure a new project and get status information for a existing project.
- **User**. This actor represents all the roles of participants in a software development project: project

manager, integrator, analyst, designer, etc. Through this actor students can perform project development and management tasks.

- **Supervisor**. This actor represents the professor of the course. Supervisors use the application to assign case studies to teams and to monitor project progress.
- **Administrator**. This actor represents the users in charge of system administration and monitoring tasks.

Figures 4 and 5 show, as an example, two screen snapshots of the APIS graphical user interface for two different actors.

B. Non-functional requierements

In this section we describe some additional requirements on APIS that are not functions intended to solve users needs but rather technical constraints and quality factors.

1) Security

Some primary requirements related to security have to do with access control mechanisms (authentication and authorization) and session management. In this sense, we have implemented a role-based access control (RBAC) scheme. This approach is based on the organization of the users of the system in a hierarchy of roles and the mapping of systems functions to roles. This way each user can perform an operation only if that operation is authorized for the current, authenticated role of the user.

The authenticator manager of the application makes use of the LDAP server of the UPM, which has a centralized identity server for students and staff and enforces the common policies for access control at University level. This way APIS does not need to store passwords or credentials, making the application more secure and easy to maintain and upgrade.

At the level of session management, requests in open sessions are validated twice in order to check session validity and to avoid session hijacking. Hijacking happens when a malicious user steals the session identifier to gain unauthorized access to the system.

2) Usability

Usability has to do with the ease of use and learnability of a software system and thus it is of particular importance for an application intended to support a learning process.

The design process of the APIS application has put special care in the definition of user needs and the user interaction patterns needed by users to fulfill these needs: data capture, navigation, document upload, presentation, etc.

APIS also provides general and contextual on-line help to give assistance to the users as well as tutorials, academic materials, technical documents on software engineering topics and document templates for project artifacts and deliverables.

3) Maintainability

This quality factor has to do with the ease with which a software system can be maintained in order to correct defects, incorporate new functionalities or cope with a changing environment.

This requirement is also very important in general and, in particular, when the system is not developed by a software company with a mature process framework and a powerful development environment and quality assurance process. This is the case for an academic development that is carried out by students who leave the university after finishing their studies and are replaced by new students pursuing their MSc degree. The new students must be able to understand and evolve systems that were initially developed by former students that are not available any longer. In this context, the design of the APIS application has put the focus on maintainability by providing technical documents detailing the system specification and design and documenting the source code as well as setting up an automated software configuration management system to keep control of system versions and change requests.

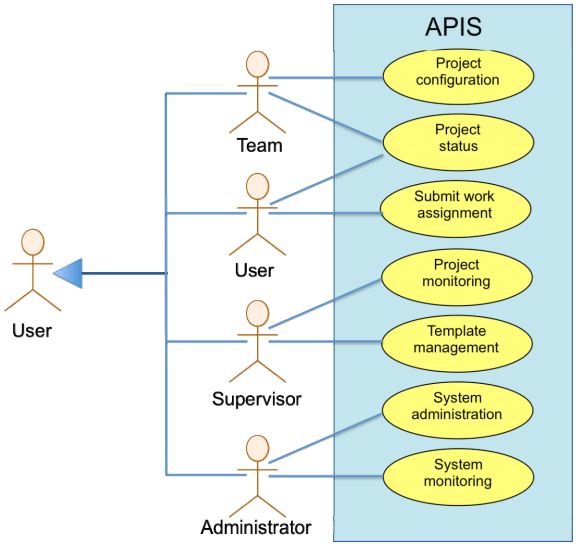

Figure 3. Use cases diagram for APIS

Figure 4. APIS graphical user interface. Supervisor view

Figure 5. APIS graphical user interface. Team view

C. Architecture and implementation technologies

APIs is a Web-based distributed software system with a high level system design (architectural design) following the so-called three-tier client-server architecture. This means that the systems is structured in three independent subsystems that may be deployed in several computing nodes in a network.

These subsystems are the following:

- **Presentation**. Components in this subsystem implement the application interaction model with the user and the presentation of output information for the user. APIS uses a Web browser and other components that allow the dynamic generation of web pages and interactive web applications.
- **Logic**. This is the subsystem controlling the whole application, i.e. the application logic.
- **Data**. This is the subsystem interfacing with the persistent data store of the application (e.g. a database).

Apart from the usual advantages of modular software with well-defined interfaces, the three-tier architecture is intended to allow any of the three tiers to be upgraded or replaced independently as requirements or technology change.

The following list shows a summary of the technologies used to implement the APIS application.

- Java platform (J2EE). Programming and execution environment.
- Apache Struts. Framework for the development of Java Web applications.
- Apache iBatis. Persistence framework which automates the mapping between SQL databases and implementation objects
- Javascript and AJAX. Scripting languages for interactive Web applications.
- HTML y CSS. Data format and presentations style sheets
- Apache Tomcat. Execution container for Java components.
- MySQL. Relational database management system.

D. System testing

In order to confirm that the implemented functionality meets specifications and that it fulfills its intended purpose we have designed and executed a big number of test cases on individual system components and on the system as a whole. In this sense, we have carried out unit tests, integration tests, stress tests (simulating many concurrent users interacting with the application) and system tests (alpha and beta tests).

These tests provided a reasonable level of confidence about the system fulfilling functional and non-functional requirements and satisfying user needs and expectations.

VI. CONCLUSIONS

In this paper we have summarized our experience of educational innovation around the software engineering learning process in the MSc degree in Telecommunications Engineering at the Universidad Politécnica de Madrid. In particular, we have described a project for the adaptation of the Software Engineering Lab course to the new educational and organizational guidelines coming from the so-called Bologna convergence process in Europe. In this context, we have described in detail the purpose of the pilot project, the organization and educational methodology of the course, the tool environment supporting the learning process and APIS, a Web-based tool supporting the collaborative, project-based learning in software engineering.

The experience has allowed us to revise and improve the efficiency of the learning process in terms of course organization, educational methodology, tool environment supporting the process, students motivation and course results, not only from the point of view of scores but in terms of the acquisition of new knowledge, skills and abilities.

This judgment is based on observation and assessment of course results by the professors involved in the course and also on the various evaluation questionnaires fulfilled by course students. We set our own specific questionnaire to evaluate specific items related to the educational innovation pilot project and additionally another University-wide general questionnaire is passed to students every year at the end of each semester. This general evaluation process has been redesigned in the 2009-10 academic year to conform the general guidelines of the Spanish Agency for Quality Assurance and Accreditation in Higher Education (ANECA). The results of all the questionnaires have been downright positives and encouraging.

Besides the evaluation of course results and students satisfaction, we had a specific interest in measuring the needed dedication (workload) of students to pass the course in terms of quantified effort in ECTS. This resulted in an estimation of around 3.7-4.4 ECTS for the course, depending on the hours of study corresponding to one ECTS (25-30). From these results, the equivalent effort for the professors in the course is about 50-55 hours per course.

We have quite a few remaining challenges ahead to keep the course and discipline evolving in the right direction to meet the requirements of a changing and demanding environment, but maybe the most important one is the adaptation of the discipline in the context of the new structure for the Telecommunication Engineering degree in terms of a Bachelor plus a Master degrees imposed by the Spanish translation of the European directives regarding the Bologna convergence process.

ACKNOWLEDGMENT

The authors would like to acknowledge the contribution and dedication to this project of Mercedes Garijo and Miguel Angel de Miguel, Software Engineering Lab professors, who have shared their classrooms and ideas with us over the years.

We would like also to acknowledge the debt we owe to Pedro Clemente and Luis Mateos, former students and authors of the fist version of APIS.

Last but not least, our thanks to the UPM's vice chancellorship for academic organization and strategic planning for their support and funding of this educational innovation project.

REFERENCES

[1] Confederation of EU Rectors' Conferences and the Association of European Universities (CRE). The Bologna Declaration on the

European Space for Higher Education: an Explanation. http://ec.europa.eu/education/policies/educ/bologna/bologna.pdf

[2] C. Larman, Applying UML and Patterns. An introduction to Object-Oriented Analysis and Design and Iterative Development, 3ª edición, Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004.

[3] W. Appelt. WWW based Collaboration with the BSCW System. SOFSEM'99: Theory and Practice of Informatics. Lecture Notes in Computer Science, 1999, Volume 1725/1999, 762, doi:10.1007/3-540-47849-3_4.

[4] ISO/IEC 12207:2008 "Systems and software engineering – Software life cycle processes," ISO JTC 1/SC 7, Mar. 2008.

[5] A. Abran, J. W. Moore. Eds "Guide to the Software Engineering Body of Knowledge (SWEBOK)" IEEE computer Society, Mar. 2004.

[6] J.T. Marchewka. Information Technology Project Management. Providing Measureable Organizational Value. 3ª Edición. John Wiley & Sons, 2009.

[7] R. Perez-Rodriguez, M. Caeiro-Rodriguez, L. Anido-Rifon. Adding Process-driven collaboration support in Moodle. Frontiers in Education Conference, 2009. FIE'09. 18-21 Oct. 2009.

AUTHORS

Juan C. Yelmo is with the Department of Telematics Engineering at the Universidad Politécnica de Madrid, Spain (e-mail: jcyelmo@ dit.upm.es).

Juan Fernández-Corugedo was with the Department of Telematics Engineering at the Universidad Politécnica de Madrid. He is now with Medianet Software, Madrid, Spain. (e-mail: jfcorugedo@yahoo.com).

This article is an extended version of a paper presented at the IEEE EDUCON20211 Conference, held from April 4th-6th, 2011, in Amman, Jordan. This paper is the **winner of the EDUCON2011 Best Paper Award**. Received, May, 3rd, 2011. Published as resubmitted by the authors May 16th, 2011.