Application of Software Engineering Techniques to the Advanced Laboratory of the Cooperative Course

Becerra, Jorge L. Risco [1], Melnikoff, Selma S. Shimizu[2], Lobo, J. Eduardo Morello [3], Borsoi, Beatriz T. [4]

***Abstract* – Practical Software Engineering courses are generally dealt with software system development. Students apply the waterfall life cycle to realize their project. The environment which supports the course is rigid and it does not allow to explore variability of software processes and techniques. This work presents one solution utilized in Software Engineering Advanced Laboratory of Computing Engineering Course of Escola Politécnica of University of São Paulo. Its environment has been conceived to offer to the students at the last degree of course a project environment based on Capacity Maturity Model and Unified Process. The project teams are organized into software factories and the students have opportunity to simulate real project environments.**

Index Terms – software engineering techniques, software processes, software education, didactic laboratory, software factory.

INTRODUCTION

The engineering courses of Escola Politécnica of University of São Paulo (EPUSP) have two-year basic cycle, before the division of student onto the respective courses. From the third year, the cooperative course students begin to alternate academic and full time trainee modules every four months, until the course accomplishment. The four trainee modules lead the students to gain early technical knowledge and practices. The academic modules formalize the learning with concepts and new techniques which allow the students to complete their corporative experience.

The software track in the Computing Engineering course includes the following courses: programming laboratory, where the students learn algorithm and data structures; object-oriented programming laboratory; where they learn programming techniques; software engineering, where they learn basic concepts; Software Engineering Basic and Advanced Laboratories; software quality and software project management. The Basic Laboratory focuses software engineering techniques applied to small size software development. The Advanced Laboratory focuses middle size software development and project management techniques.

To earn Software Engineering learning, the students should practice techniques to develop software systems in an environment where they can select tools and solutions from different options and suppliers. In this context, the lecturers should supervise project activities to conduct software engineering teaching and should play the role of project managers.

The Software Engineering Advanced Laboratory has been structured based on Unified Process and CMM (Capacity Maturity Model). Its processes are represented through BPMN (Business Process Modeling Notation) and several tools have been used, including a web based project portal.

The students improve their knowledge in analysis, design, coding and testing obtained in previous courses. They are presented to new concepts like resource estimation, software architecture, reuse technique, design patterns, configuration management. The project team is organized following the concept of software factory.

The environment of Advanced Laboratory has been conceived to be configurable to several projects to be developed into it.

Although this paper considers didactic techniques, like the cycle of experimental learning [10], the emphasis is centered in the software development environment and an application of this environment by one specific team and in one specific project.

THE SOFTWARE DEVELOPMENT ENVIRONMENT

The Advanced Laboratory environment, in which the activities of software development are carried through, is based on the concept of software factory [1]. The software factory model is used as reference model for the definition of development environment elements: the software processes (included management), the interface between different processes, software activities, documents, tools and roles.

The software factory model facilitates the management of adaptability and flexibility of the environment because of software techniques selected, educational strategic, specific software system to be developed. This model is composed by views, where each view represents groups of requirements of the elements of the environment. Thus the views guided the definition of software environment elements.

The software factory model is constituted by three views: organizational view which represents strategic, management, and operational levels; process view which contains process models (static and dynamic models) and relationships between them; infrastructure view which contains technologies and tools which support the activities

[1] Becerra, Jorge Luis Risco, Escola Politécnica of University of São Paulo - Brazil, jorge.becerra@poli.usp.br
[2] Melnikoff, Selma Shin Shimizu, Escola Politécnica of University of São Paulo - Brazil, selma.melnikoff@poli.usp.br
[3] Lobo, José Eduardo Morello, Escola Politécnica of University of São Paulo - Brazil, eduardo.lobo@poli.usp.br
[4] Borsoi, Beatriz Terezinha, Escola Politécnica of University of São Paulo - Brazil, beatrizborsoi@terra.com.br

of software development and management. Table I presents the major input and output elements of the views.

TABLE I
VIEWS, INPUTS AND OUTPUTS

	Organizational view	Process view	Infrastructure view
Input	Existing software factories roles. Abilities and knowledge of teams.	Models and standards of quality. Type of project to be developed. Techniques to be learned.	Technological resources of different suppliers (Microsoft and IBM). Type of activities (manual and automatic activities).
Output	Organizational model of teams.	BPMN process model (roles, activities, devices and resources). Document Standards.	Technological architecture of the environment.

The process views is defined based on ISO/IEC 12207 Standard [2], Capacity Maturity Model (CMM) [3], Unified Process [4], Software Engineering techniques [5] and the model of software factory created in the Technology of Software Laboratory of EPUSP (available in http://143.107.170.130:5108).

The students are organized into four teams, and each team is a software factory with eight students. Two factories developed activities in one day of week and others in other day of week. The constitution of each team is realized by the students themselves.

The strategic management team is constituted by one lecturer and two doctor degree students. The project management team is constituted by four students, one of them from each factory. Each factory has its own responsibility. The product integration factory has the responsibility to develop concept proofs and to integrate software modules produced by other factories. The business logic factory is responsible for requirement elicitation and analysis. The human-computer factory and the data base factory are responsible to select development tools and environments and to develop interface and database subsystems. In the implementation phase all factories participate in programming activities.

The target system developed in the last period was an ERP (Enterprise Resource Planning) system for a beer enterprise.

The software factories received the system requirements including technology constrain, like as .Net, Java and Web services. The subsystems were defined and its modules developed by factories which delivered tested modules.

Many tools were used during the system development: Java and .Net frameworks as development tools; RUP (Rational Unified Process) as process framework; UML (Unified Modeling Language) as modeling tool; SourceSafe as configuration management tool; Microsoft SharePoint as portal construction tool;

The project portal is a document repository of factory members. It has a public area, which can be accessed by any factory member, and private areas, one for each factory. In private area, each member has his own private folder.

During the project, the students have been supported by one technician to solve network and software configuration problems. The strategic management team followed the student activities, supporting the students with technical and managerial knowledge. This team performed also didactical supervision, evaluating the student learning growing and project evolution. The student evaluation is carried out through individually produced artifacts, set of final documents and evaluation of ERP system.

THE ENGINEERING TECHNIQUES APPLIED TO THE LABORATORY

During the project, many techniques are used: work group technique, techniques for negotiation with customer, communication techniques inside and between the factories, meeting techniques. Partial and final results are presented to customer and to strategic management team for project and learning evaluation.

During requirements elicitation phase, interview techniques and analysis of similar systems are used. The students interviewed the strategic management team members, which assumed the role of customers. During process modeling phase, BPMN [6] is used. For analysis and design phases, UML static and dynamic diagrams [7] are used. The software architecture design [8][9] began during analysis phase and has finished at beginning of implementation phase. This strategy allows the architecture design in the evolutionary manner through the development process.

During system implementation phase, prototyping techniques are used as concept proofs. These proofs included .Net framework utilization for front-end system, Java language for back-end and web services for the final system implementation. For testing the following test types are used: unitary tests, modular tests, integration tests and product acceptance tests by customers.

REAL PROJECT

In the last developed project, the students have received at the beginning of this course the task of developing an ERP based on web service. The subsystems of this system were distributed to four software factories and at the end of each subsystem project the students had to integrate their products with products of other factories.

The project carries out an important function in the didactic environment because its complexity, its development form and its business-oriented characteristics create disturbances in the education aspects. In this context, the students have the opportunity to apply: the basic concepts of software engineering, different project techniques, programming techniques, network computing, economy, administration and other knowledge acquired during the Computing Engineering course. The project creates new activities to lecturers: revision of development processes, configuration of the software engineering laboratory

environment and application of several techniques of education and pedagogical evaluation.

The real project offers the opportunity to apply didactic, software engineering and complementary techniques, simulating the real environment in which the students will be working in their future.

The project changes and impacts the development and education aspects of the software engineering laboratory environment.

RESULTS

Table II presents some problems and solutions met in the Advanced Laboratory; looking for solutions to these problems, the students learn practices about software development. Left column presents main problems occurred in laboratory; centre column presents solution; and right column presents environment support to problem solution.

TABLE II
PROBLEMS AND SOLUTIONS

Problem	Solution	Environment support
Communication between factories	Systematic use of email and exchange of electronic messages	Exchange server SharePoint portal SharePoint to cooperative portal
Need of decision in real time	Meetings during the development. Use of a file of meeting summaries	Project repository
Need of technical decision of data base type	Meeting of teams to present the technical criteria. Presentation of the report about the solution	Standard documents to facilitate the analysis of technology
Module integration	Integration plan based on software architecture	Integration activities established in process models

Relating to communication between factories, the project development environment and its infrastructure provided the network to establish better communication among students . The solution allowed faster decisions to project activities, usually required in a real corporative environment.

Relating to project decision in real time, the students had to exercise some techniques and to develop administrative skills to manage the available time and resources. The project repository environment allowed reliable information to any project member team in real time.

Relating to the decision of data base type, standardization defined by factory teams and the standardized documents facilitated the finding and analyzing project information between various software factories.

Finally, the ERP system modules integration usually is a biggest problem of the software factory. The process architecture has facilitated this integration through the standardization of documents, the technology and correct interface between processes.

CONCLUSION

The organization of the Advanced Laboratory contributes to simulate the real world of software development. Its view based organization facilitates the composition of a very flexible environment to realize several software engineering teaching experiments.

The processes which define the Advanced Laboratory have been specified and modeled using BPM techniques. Thus, the insertion of a new software technique in the environment requires changes in the models of visions, modification in development processes, and reviewing of the procedure about software engineering education.

The Software Engineering Advanced Laboratory environment can be replicated in other engineering courses, because its elements, views and processes are independent of educational procedure, technology and software process. For this purpose it is necessary to modify the models of the views (organizational, processes and infrastructure views), according to technological and educational requirements of the course.

REFERENCES

[1] R. Panigassi, “Método para definição e modelagem de processos de fábrica de software usando RM-ORP e BPM”. Dissertação (mestrado em Engenharia Elétrica) - Universidade de São Paulo, 2007, pp. 158.

[2] ISO/IEC The International Organization for Standardization / The International Eletrotechnical Comission “ISO/IEC 12207 Information Technology – Software Life Cycle Processes”, 1995; Amendement 1, 2002; Amendement 2, 2004.

[3] CMM. “The capability maturity model: guidelines for improving the software process”, 14th ed., Addison Wesley Longman, 2000, pp. 441.

[4] I. Jacobson, G. Booch, and J. Rumbaugh, “The unified software development process”, Addison Wesley, 1999, pp. 528.

[5] R. S. Pressman “Software engineering – A practitional approach”, 5th ed., McGraw Hill, 2001, pp. 860.

[6] S. A. White, “Business process management notation”, 2004, pp. 296.

[7] G. Booch, J., Rumbaugh and I. Jacobson, “UML. Guia do usuário”, 13rd ed., Rio de Janeiro: Elsevier, 2000, pp. 550.

[8] S. T. Albin, “The art of software architecture: design methods and techniques”, John Wiley & Sons, 2003, pp. 312.

[9] L. Bass, P., Clements, and R. Kazman, “Software architecture in practice”, 2nd ed. Addison Wesley, 2003, pp. 560.

[10] Kolb, D. A., *Experimental Learning – Experience as the Source of Learning and Development*, Prentice-Hall, New Jersey, 1984.