Revista Electrónica de Investigación Educativa

Vol. 3, No. 2, 2001

Application of Process Modeling in a Software- Engineering Course

Aplicación del modelado de procesos en un curso de ingeniería de software

Gabriel Alberto García Mireles
gagmireles@hotmail.com

Jaime Nunó 2680
Ampliación Hidalgo, 22880
Ensenada, Baja California, México

Josefina Rodríguez Jacobo
jacobo@cicese.mx

Departamento de Ciencias de la Computación
Centro de Investigación Científica y de Educación
Superior de Ensenada (CICESE)

Km. 107 Carretera Tijuana-Ensenada
Ensenada, Baja California, México

(Received: August 8, 2001; accepted for publishing: August 28, 2001)

Abstract

Coordination in a software development project is a critical issue in delivering a successful software product, within the constraints of time, functionality and budget agreed upon with the customer. One of the strategies for approaching this problem consists in the use of process modeling to document, evaluate, and redesign the software development process. The appraisal of the projects done in the Engineering and Methodology course of a program given at the Ensenada Center of Scientific Research and Higher Education (CICESE), from a process perspective, facilitated the identification of strengths and weaknesses in the development process used. This paper presents the evaluation of the practical portion of the course, the improvements made, and the preliminary results of using the process approach in the analysis phase of a software-development project.

Key words: Software engineering, software-development process, process modeling, teaching project-based software engineering.

Resumen

La coordinación en un proyecto de desarrollo de software es un factor crítico para liberar un producto de calidad dentro de las restricciones de tiempo, funcionalidad y costo acordadas con el cliente. Una de las estrategias para abordar este problema consiste en utilizar técnicas de modelado de procesos para capturar, evaluar y rediseñar el proceso de desarrollo de software. La valoración de los proyectos realizados en el curso de ingeniería y metodología de la programación impartido en el CICESE, desde la perspectiva de procesos, facilita la especificación de fortalezas y debilidades del proceso de desarrollo utilizado. Se presenta la evaluación de la parte práctica del curso, las acciones de mejoramiento llevadas a cabo y los resultados preliminares al utilizar el enfoque de procesos en la etapa de análisis de un proyecto de desarrollo de software.

Palabras clave: Ingeniería de software, proceso de desarrollo de software, modelado de procesos, enseñanza de la ingeniería de software basada en proyectos.

Introduction

The purpose of software engineering is to generate and maintain software systems within the constraints of time, functionality and costs agreed upon with the client. The goals of this technological discipline are to improve the quality of products developed and to increase the productivity of software engineers. The degree of formality and the time allocated to the software project vary according to the size and complexity of the product to be developed.

As the complexity and size of the project increases, coordination becomes more difficult due to increased communication among the software engineers, managers and customers (Fairley, 1985, Kraut, and Streeter, 1995). In the educational arena, research indicates that graduates of bachelor's programs have little understanding of the meaning of large-scale programming, that is, applying the principles of software engineering to product development by a team of people, during which time the effective coordination between the participants allows the development of a successful software system (Upchurch and Sims-Knight, 1998).

Various strategies are used in teaching software engineering. Some of these are based on a review of literature, without putting into practice the knowledge gained from carrying out a project (Tomayko, 1987). Other strategies are based on teamwork; their main objective is for the student to experience the development of a product for a real client, to make decisions according to the options or resources available, and to face the issues of communication and coordination typical of group work (Upchurch and Sims-Knight, 1997).

A recurring problem is that the success of the projects in these courses depends on the instructor's skill and experience in directing projects. It is probable that different instructors will achieve different results using the same model; or that the same instructor, with another group of students, will get mixed results. Furthermore, in some of these courses attention is paid only to the characteristics of good architecture and implementation, without incorporating the aspects of quality assurance and administration. These problems suggest that many software-engineering projects suffer from deficiencies in the software-development process used by the instructor (Collofello, Kanko, and Kantipundi, 1994).

The software-development process can be defined as the set of activities, methods, practices and transformations that individuals use to develop and maintain software and associated products (Paulk, Weber, Curtis, and Chrisis, 1995). A defined and effective process reduces the effort in developing a software product, and increases the productivity of the development team (Clark, 2000). Indeed, the modeling and execution of the software-development process constitute, in software engineering, a major research area (Maurer and Kaiser, 1998) whose purpose is to propose solutions for problems in the organizational context based on the exploitation of technologies for coordination and integration (Warboys, Kawalek, Robertson, and Greenwood, 1999).

In the search for excellence in teaching process-based software engineering, various approaches have been used. For example, in certain courses, students are given a requirements-specification document. The purpose is to do the work specified, covering all the stages of product development, starting out by following certain processes described in a textual manner (Upchurch and Sims-Knight, 1998). Other researchers have modeled the software-development process using object-oriented techniques, with the aim of understanding the complexity of the process step by step (Oktaba and Ibargüengoitia, 1998). Also, there are used standards and practices accepted by the community as a basis for introducing the software-development process (Robillard, 1998; Jaccheri and Lago, 1997; Mayr, 1997). In addition, some experts recommend the incorporation of processes and teamwork into the design of curricula related with software engineering (Bagert, Hilburn, Hislop, Lutz, McCracken, and Mengel, 1999).

In the Ensenada Center of Scientific Research and Higher Education (CICESE), engineering and programming methodology is taught as a compulsory subject in the master's degree program in Computer Science. The features of the course allow it to be used as reference for carrying out a process-improvement project. This article describes the application of process-modeling techniques in the practice portion of thee software-engineering courses to identify the strengths and weaknesses of the development process used, and the actions to be followed to mitigate the problems detected. In the second section there is identified the methodology which served as a basis for making the process visible, evaluating it, and suggesting improvements. The third section describes the steps followed in the study of the development process, by taking as reference the steps of a process-improvement project and adapting them to the specific needs of the case study. The fourth section reports the general characteristics of the course's lab session, in which students developed a software system and summarized the deficiencies detected in the coordination of the work among the course participants. In the section are described the actions taken to resolve the problems identified. Section six presents the results obtained when the improved processes were implemented. Finally, there are described the lines followed in the course for process improvement, and the conclusions of the work.

I. Modeling the software-development process

The methodology used to improve the process followed in the software-engineering course considers the stages of definition, capture, evaluation, redesign and execution (Wastell, White, and Kawalek 1994; Caputo, 1998, Arthur, 1992; Sommerville, 1995; Warboys *et al*., 1999). The definition phase establishes the objectives of a process, delimits the boundaries, shows the main entries and exits, indicates the customers who would be benefited; and the providers of entries. The representation and capture phase models the process in detail, based on the information obtained from interviews, document revision and plans for generating a graphic image of the process. The evaluation phase analyzes and evaluates the process with the overall purpose of seeking weaknesses and problems relating to ineffectiveness or inefficiency in achieving the goals of the established process. The results of the evaluation facilitate the redesign of the process, which uses a modeling language comprehensible to process users. The redesigned process was set in motion in the organization to verify that it truly met the established goals.

In the process modeling, four aspects are considered: functional, performance, organizational and informational (Curtis, Kellner, and Over, 1992). In the functional aspect are contemplated process activities that were being executed, plus the flow of the most relevant matters (documents). In the aspect of behavior or performance attention is paid to the time in which activities were carried out (conditions, sequence and iteration). The organizational view of the process focuses on the physical place within the organization where the activities took place and the person who had the responsibility of effecting them. Finally, the informational aspect deals with the provision of documents in the coordination and communication among the functionaries.

The capture of a process can take many forms, and a graphic language can be used to present it. The description of a process can take many forms, and a graphic language can be used to present it. Graphic models help to illustrate the process under study by reducing complexity, promoting a common understanding among the participants and allowing the study of alternatives (Miers, 1996); they can also be used to influence, control and direct what happens in the real world (Warboys et al. 1999). In organizational settings, they permit the capture of the process behavior to be analyzed later, and also act as repositories for organizational knowledge in order to facilitate learning about the organization and its processes (Ould, 1995).

The technique for preparing the diagrams of software-development process in the study was the Role Activity Diagram (RAD). This technique describes the process from the standpoint of roles (Ould, 1995). Those in the roles implement activities and make decisions in accordance with the rules of the organization; they may perform parallel activities and interact with other functions as the work progresses in order to achieve the goals of the process. The actions may include the use or production of information or documents. In most cases, the modeling process leads to the immediate identification of options for redesign (Miers, 1996).

The success of the project in the course depended on the cooperation achieved among the students, while coordinating the efforts of each of the participants according to his/her position; this was essential for meeting the goals of the course. The study of interactions, activities, places and documents produced while working on the project could improve the software-development process.

II. Work method

The objective of this study was to systematize the development process used so far in the engineering and methodology-programming course, to identify useful practices and determine the weaknesses of the process with the aim of improving it. Although there is no unique approach to solving the problems of software engineering, process improvement was presented as an alternative, to increase the productivity of software engineers and to generate higher quality products (Hersleb, Zubrow, Goldenson, Hayes and Paulk, 1997; Clark 2000). In improving the process followed in the course, the goals of software engineering were considered (Fairley, 1985; Pressman, 1993) so as to release a product with the functionality agreed upon with the client, a product with the required quality and within the time stipulated for the project. Thus, in the course, a successful software-development project was one that met the goals outlined. For this study, we took as reference the stages of a process-improvement project. Table I summarizes the application of this framework to the particular environment of the software-engineering course.

Table I. Stages in the study of the software-development process and improvement of the activities of the analysis phase of the project

Stage	Purpose
1. Definition	To determine the current state of the software-development process from the perspective of each participant, and propose improvements in the tasks related to systems analysis.
2. Representation	To analyze the documents of previous projects so as to learn about the contributions of each participant to the software-development process. To carry out semi-structured interviews of students who worked on the project (described in the preceding paragraph) with the aim of supplementing and verifying the activities, relationships with other participants, problems detected and suggestions for improvements. To review notes on the theory of the course so as to determine the scope of the issues related with systems analysis.
3. Presentation	To facilitate the communication and comprehension of the process through graphic models of the overall development process of development and the activities of each agent. The information from the previous stage was considered for obtaining the models.
4. Evaluation	To identify weaknesses in the systems analysis phase of the development process by taking the current model as reference, comparing the reports of the representation stage (results of the interviews and assessment of the course notes) and comparing this information with current literature on the theme.
5. Redesign	To create a process proposal for the systems-analysis phase, considering the results of the previous section (evaluation).
6. Execution	To use the models generated in a new project. This activity requires training the students to use the models.
7. Evaluation	To determine the impact of the use of models and standards in the software-engineering course.

The definition stage deals with the software-development process as an integrated unit of the activities of quality assurance, and of product management and engineering. The study of the development process begins with each student's being assigned a position in the project, and ends when a functional product is delivered to the customer. In the first stage of the study, the current status was determined, and models of the activities and interactions of each of the agents were generated. In the second stage, we evaluated, suggested improvements and carried out the development process, using as reference the models generated in the redesign of the systems-analysis phase.

The capture and representation stage aims to know the current status, for which are used various sources of information. In the particular case of this study, documents generated during the projects of previous courses, interviews with participants in those projects were considered, and the course notes were reviewed:

a. Documentation of earlier projects. Since 1994 the subject Engineering Methodology and Programming has been taught at the CICESE. The projects chosen were DASIS (January to April) and GENSIS98 (September to December). Both were carried out in 1998 by first-year MBA students. The decision to do such projects as the basis for modeling the development process was made because the information generated is in an accessible electronic-information repository. The documents are organized by agents, and are those which are better documented. This is because, unlike documents from previous projects, these contain the profile information of each agent in the software project, as well as the documents generated during the project's life cycle. Another factor considered was the similarity between the projects undertaken, since the development platform, analysis and design methods, and types of technical revisions were the same. Furthermore, the first author participated as a student in the DASIS project, and the second, as an auxiliary in the development process. The details of the projects were classified as: 1) strategic planning (mission, vision, values, rules), an exercise that took place early in the course; 2) description of the profile of each agent (goals, activities, relationship with other participants, support tools and bibliography); and 3) documents associated with the project (requirements, software specification, design, code modules, user manual, work plans for each of the functions, minutes of technical meetings, reports of activity progress, etc.).

b. Semi-structured interviews. A script was prepared for interviewing the students who were working on the ENSIS98 or DASIS projects. The analysis of the profile of each agent was considered (letter a), with information based on the literature of this technological discipline rather than on practical experience (this theoretical research was conducted by students at the beginning of the course). Of the 28 students (14 in each project), a sample of 10 students, each having a different job, was selected. In this way, observations were made for each of the 10 positions available on the course. The interviews were conducted from July 23 to July 29, 1999. The duration of each was approximately an hour. In the interviews were explored the activities that were actually performed, their sequence, the documents generated, the tools used, and the interactions with other peers, so as to verify and complete the information described in the profile of each agent. Also, the interviewees were asked about the problems they faced in carrying out their activities, as well as their suggestions for improving the laboratory portion of the course.

c. Theoretical documentation of the course. In this area, the material used in the theory of the course, the bibliographic references and sequence of topics were examined, in order to verify that the theoretical portion had to do with the topics necessary for carrying out the laboratory activities of the course. The subject reviewed in greatest detail was that of systems analysis, since this was the basis for the redesigning of processes in this study.

The capture and representation of the process was based on the analysis described in previous paragraphs, and permitted the generation of several models for understanding the current development process. The support diagrams used to focus on the different process perspectives were: the rich image, whose purpose is to give a detailed picture of the problem (Warboys *et al*., 1999; Wastell et al., 1996); the general diagram of the software-development process, in which are grouped together the activities in the areas of administration, control and development (Donaldson and Seigel, 1997); the state transition diagram, to identify the dynamic behavior of the process; and the parent role activity document with the aim of identifying the responsibilities of each position, the relations between them, and the documents generated or used in each of the activities.

RAD diagrams were represented in the refined information on the activities effected by each of the agents, the interaction between them, and the documents or products generated during the project's life cycle. The approach to representation of the process took as reference the activities each agent or position executed (Ould, 1995; Kawalek, 1994).

The process evaluation and redesign focused on the activities done at the analysis stage of the software-project life cycle. The reason for dealing with this stage is that since it is the first phase of the project, students are not yet familiar with the process used. Furthermore, a study indicates that much time was spent on this phase of the project (34% on DASIS and 46% on GENSIS98), and coding and testing activities of the system were neglected (Garcia, Rodriguez, Mireles, and James, 2000).

To evaluate the analysis process, the basic bibliography of software engineering (Sommerville, 1995; Pressman, 1993; Fairley, 1985) and specialized literature (Dorfman and Thayer, 1990; Hare, 1992) were taken as references. Of particular interest was the review of issues associated with the preparation of questionnaires for the interviews, the identification and classification of requirements, and the content and format of documents of customer requirements and software specification. In addition, we analyzed the themes related to the administration of these, as well as quality criteria applied at this stage of the project's life cycle, and the relationship of those needs with the planning of the project and the administration of the configuration.

The process redesign was a result of the previous stage, and in it the information was presented in RAD diagrams. In this phase of the process-improvement project, there was taken as reference the analysis stage of the traditional life-cycle ("cascade") of a software project, divided into three parts, according to the landmarks of this project phase and according to the stages of requirements management: production of the requirements document, validation of these by the client and development of the software specification.

The new models generated for the system analysis stage were presented to two students involved in these activities; the students occupied the positions of analyst and quality-control engineer in the projects evaluated. There were informal talks with them in late August, 1999, for approximately 30 minutes. The talks were designed to validate the clarity and complete redesign of the process. The purpose of the model and the meaning of each graphic element were explained to the students. Based on their perception of the development process, they indicated that the model was adequate.

The purpose of generating models is to use them to execute the process. The improvements in the analysis process were applied in the period from September to December, 1999, during the engineering and programming methodology course, with 15 beginning students of the master's program. There was a class on the software-development process, and the models were given to all the students so that they could become familiar with activities related to systems analysis. After class, informal interviews were conducted with three of the students directly involved with the activities of this phase (requirements engineer, system architect and quality-control engineer). At the end of the course an opinion questionnaire was prepared, and the participants responsible for analyzing the course were interviewed. The purpose was to find out what impact the process approach had had in the course on Program Engineering and Methodology.

III. Course description

The objective of the Program Engineering and Methodology course was "to understand the development of software projects of medium scale (14 to 20 people), evaluate the procedures, combine tools, define processes and build the organizational memory" (Licea, Rodriguez, and Favela, 1996), with the aim of improving the software-development process. The strategy used was an adaptation of the model presented by Tomayko (1987), in which was described the development of a software project for a real client, and every student in the class played a role or had a position in the software-development process.

Part of the course was theoretical, part, practical. The purpose of the theoretical portion was to show the activities involved in developing a software system and the relationship between product engineering, quality assurance, and project management. It also analyzed the methods and techniques that exist today in each of the disciplines of software engineering.

In the practical portion, students formed a "company" to develop a software system. The instructor served as a consultant. The students, together with the instructor, determined the company name and logo which identified them. They worked on the definition of the group's mission, vision, values and rules to be followed for the duration of the course. Each student occupied a position in the company; this was determined by means of a job interview with the instructor; in the interview the knowledge and skills of each participant were evaluated. The positions available were: project manager, quality control engineer, engineering, validation and verification engineer, documentation specialist, configuration manager, analyst, designer, programmer, test engineer and maintenance engineer.

Also in the practical portion, students held weekly technical-review meetings governed by an agenda. In these meetings work progress was reviewed, and decisions were made regarding the quality of the documents presented, problems faced in the development of the system, and the impact of their decisions on the task calendar. During the technical reviews, some students acted as reviewers, and the rest, as process observers. At the end of the review, the instructor gave suggestions to students regarding the activities to be performed to correct the problems detected.

The software project associated with the course had a real customer, who would validate the system when it was released. However, the project had to be adjusted to certain restrictions implicit in the objectives and environment of the software-engineering course (Garcia and Rodriguez, 2000):

- The delivery date for the system developed was governed by the school calendar (approximately 3 months).
- Students were taking other courses (an average of three) while working on the project.
- Some students had not previously attended courses related to this technological discipline.
- The methodology of analysis and object-oriented design used was the Unified Modeling Language (UML).
- The development platform for the software system was the World Wide Web (www), so that it was necessary to become familiar with the programming languages focused on this platform.
- Participants were required to generate a centralized information repository on the www platform.
- In addition, facilities were provided for the use of Email as communication support.

The main goals of this course required students to learn to work together; understand the relationship between product engineering, quality assurance activities and administration; to know the methodologies, techniques and latest tools in software-development systems and apply them in solving a real problem. A very important point was that the entire group should participate in the development of the project, and that no one person should carry out the software project by him/herself. This aspect of the course was reflected in the weight of the evaluation, since 50% of grade was obtained from the finished software system and validated by the functionality required by the client.

The models generated in the capture and representation stage of the process provided an understanding of the tasks carried out in projects. The activities were separated according to the typical phases of the traditional life cycle of a software project: analysis, design, coding and testing. In turn, the various positions that students could occupy were grouped into the areas of administration, control and development (Donaldson and Seigel, 1997).

In the area of administration, the purpose was to plan, organize, direct and control the tasks of the development group so as to deliver a product under the constraints of time, cost and functionality required by the client. In our study, this group of activities was performed by the person functioning as project manager.

The control group guaranteed that product integrity be achieved and maintained; furthermore, it confirmed that the development of the software had followed a disciplinary process and satisfied the customer's needs, providing visibility to the project. This group was composed of the quality-control engineer, the validation-and-verification engineer, the maintenance engineer, the configuration administrator and the documentation specialist.

The development group focused on product-engineering activities derived from the project's life-cycle stages: analysis, design, coding and testing. The positions belonging to this group were: analyst, designer, programmer and test engineer. This group carried out the key activities of the software-development process.

Next, we will briefly describe the flow of tasks performed, the interaction between the different groups and problems detected in the stages of the software-development project, according to the results of the course evaluation.

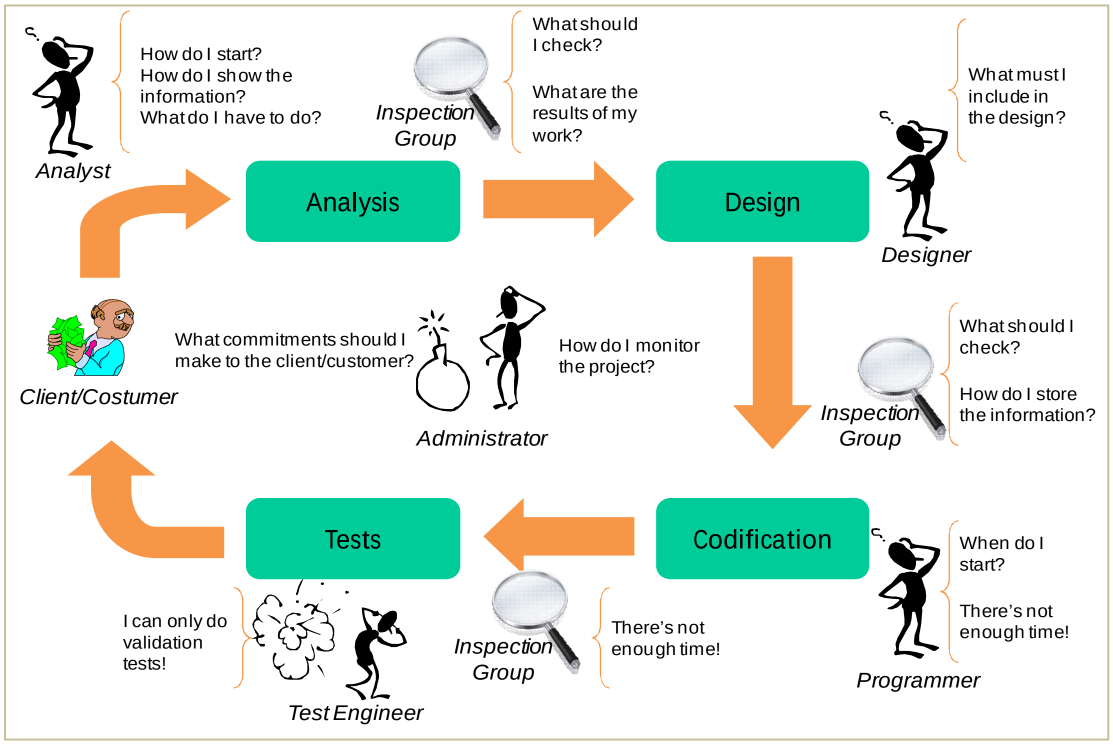

Figure 1. Rich picture of the overall software-development process

The control group's activities are embedded in the software development process; its reputation is achieved at the end of each stage, upon the evaluation and approval of the products of each stage of the project life cycle. The greatest burden of administrative work is at the beginning of the project, since that is when the administration plan and contract with the customer must be established. During the development of the system, the manager's job is to monitor the progress of the project.

Each stage of the project presents a series of challenges to the participants (see Figure 1). At the beginning of the course, students should find out what type of activities belong to their jobs and how to carry them out; define the forms and documents to be used in developing the project; and select the support tools and methodologies. The participants under pressure at this stage are the project manager and analyst, since they are the first to go into action without a clear understanding of their responsibilities (due to time constraints in the course). According to the results of the interviews, participants indicated that it would be desirable to have the support information necessary for fulfilling their responsibilities, such as the standard for the requirements document; aspects to verify in the presentation of requirements, and the structure of the project-management plan.

Many of the students had not participated before in a technical review process, in which an agenda is prepared, the issues to be discussed are defined, and a certain amount of time is allocated to each. The organization and communication between participants was low during the first sessions. Moreover, the review of a software product, which was one of the objectives of the meeting, also had its weaknesses in the earlier sessions. Although in theory the students knew the quality attributes that had to be reviewed, it was difficult for them to apply these when evaluating the documents, since the main problem they faced was to gain a clear understanding of the problem they would solve (requirements document), described by another person (the analyst).

During the second month of the course, students managed to control the logistics of the technical review, but the agents who focused on the control area still had difficulty determining the quality of the product because there are no standards that serve as a basis for that work.

In the last three weeks of the course, the group again lost control of the project. This was because they were coming to the end of their other classes, and needed more time to finish their final papers. In addition, delays in the release of the products generated during the analysis stage caused it to run over into the calendar of scheduled activities, thereby reducing the time for coding and testing activities. In the information available in the repositories of the projects, no reference was found to quality reports made during this period, nor was the configuration of the code generated even monitored.

Despite these setbacks, a prototype software system was achieved by the end of the trimester, and this was presented to the client for validation. After this session, the project manager delivered the generated documentation and source code of the prototype. Finally, an assessment of the project’s legacy was made from the perspective of each work position.

IV. Changes made in the course

The evaluation of the software process models obtained and the comparison of the functions performed by each work position, as established in the literature of this field of technology, permitted recognition of the problematic areas of process followed in the CICESE. Moreover, our interviews indicated that the lack of reliable information, the absence of standards, the lack of hardware and software resources, the excess of formal reviews and the lack of up-to-date biographical references were factors that had a negative impact on the possibility of releasing the software system within the stipulated time.

Although the model of software-development process was conducted with information from the practical portion of the software engineering course, the deficiencies detected also affected the theoretical part. Updating the theory included changing the order in which the subjects were taught, taking as a reference the stages in which the software project was carried out; and introducing the theme of process engineering, for the purpose of preparing students to understand the models that would be presented to them in the laboratory sessions.

The results of the interviews and the evaluation of the models generated showed little visibility in the analysis phase of the project. For this reason, there was a deeper exploration of the theory of systems analysis, and there were indicated the key aspects for determining the requirements and specification, as well as the management of these throughout the project's life cycle. Another theoretical aspect approached in the theory had to do with the quality attributes reviewed in the documents produced in the analysis stage, and the types of technical reviews that could be applied.

Finally, updated and classified bibliography in keeping with the study topic was provided for the purpose of presenting and analyzing the methods and techniques used in the various disciplines of software engineering. For each of the subjects, two to five key references were indicated. Also included was a list of supportive literature for the student to use as a guide in obtaining a deeper knowledge of the activities related to his/her assigned job.

In the practical portion of the course, the strategy for improvement involved redesigning the processes that had to do with the analysis stage in the project life cycle, due to the impact this phase has on the development of the software system. In order to verify the precision of the models generated, these were validated by participants in previous courses. In the new proposal, the responsibilities of the analyst were assigned to the positions of requirements engineer and systems architect. The goal of the first is to develop the requirements document, the stage in which close communication with the customer and his/her representatives is indispensable, to identify the needs that the software system must satisfy. The goal of the second is to propose a solution to the customer's needs, represented in a systems analysis language (UML).

The tasks performed during the analysis phase of the project and the interactions that took place between the participants permitted the generation of updated models for the processes of producing the requirements document, the validation of requirements, and the construction of the software system specification (see Figure 2). With the definitions of processes, students could better understand what to do, what to expect from their peers, and what they must provide. The processes are divided into roles. A role involves a set of actions carried out by an individual or a group within the organization. Furthermore, a role includes the logic that controls actions in accordance with the rules of the organization. Also, a role has the necessary resources to accomplish its activities (Ould, 1995).

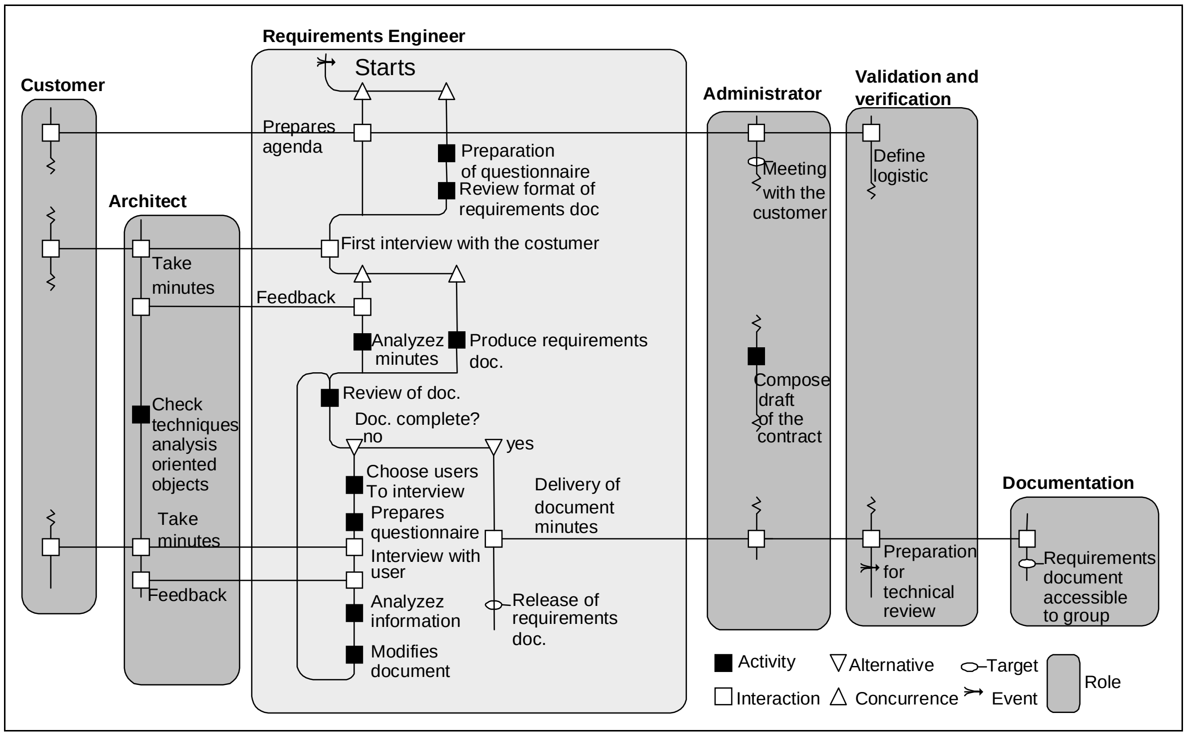
Requirements Engineer
Starts
Customer
Administrator
Validation and verification
Prepares agenda
Preparation of questionnaire
Review format of requirements doc
Meeting with the customer
Define logistic
Architect
First interview with the costumer
Take minutes
Feedback
Analyzez minutes
Produce requirements doc.
Compose draft of the contract
Review of doc.
Check techniques analysis oriented objects
Doc. complete?
no
yes
Choose users To interview
Delivery of document minutes
Documentation
Take minutes
Prepares questionnaire
Interview with user
Preparation for technical review
Requirements document accessible to group
Feedback
Analyzez information
Release of requirements doc.
Modifies document
Activity
Alternative
Target
Role
Interaction
Concurrence
Event

Figure 2. RAD diagram for the preparation of the requirements document

The principal function of the products generated (customer requirements document and software system specifications) was to facilitate communication between the customer and the development team, to guide the creation and modification of the software products, to direct the project planning and to assure the quality of the software system constructed (Rombach, 1990). The creation of these documents gave students the opportunity to model a real problem for a customer, and to deal with the issues of software product description. In doing so, the participants were introduced to the problems associated with the descriptions in natural language and the representation of product characteristics in accordance with the systems analysis methodology used (UML, in the case of these courses). During the preparation of these documents, conflicts and inconsistencies produced in communication between the client and the development team were reconciled, as well as those that occurred between the members of the software development group itself. In addition, students were forced to work on their reading and comprehension skills (Upchurch and Sims-Knight, 1998).

Another important consideration in the redesign of processes has to do with the entities (documents) that serve as input or output for the activities identified in the models (Curtis *et al*., 1992). The evaluation of the documents used in this phase of the project enabled the development of standards for the customer requirements documents and the software system specification. Moreover, guidelines were developed to conduct the first interview with the client and the list of quality attributes to be revised in the requirements document. As indicated by Sommerville (1995), the documents generated during the software development project are particularly important, since it is the only tangible way to represent the software in the early stages of its evolution. In addition, Humphrey (1989) notes that the standardization of the documents used in the development process helps to reduce the training problem and facilitates the verification of the quality criteria agreed upon.

A process-modeling project does not end with the redesign of the process, but requires training its users and evaluating the performance with the new changes. In addition to including the issue of process engineering in the theoretical part of the course, advice was also provided to the students involved in the analysis stage, to resolve their concerns regarding the use of process models and standardized documents. Without proper training, the effects of updating the process diminish its quality, rather than achieving the goal of an effective and efficient process (Sommerville, 1995).

V. Evaluation of the experience

The improvements in the model were applied during the period from September to December, 1999, in the course of Program Engineering and Methodology, which was attended by 15 beginning students in the CICESE Master's Degree Program in Computer Science. The strategies outlined in the preceding paragraph were carried out, and at the end of the trimester, a sample of three students was interviewed (requirements engineer, systems architect, and quality control engineer), since they were responsible for implementing the activities described in the redesigned models for the systems-analysis stage. During the interview, each student expressed his/her perception of the course through a questionnaire (Table II). The instrument had nine questions scored on a Likert scale (where 1 = strongly disagree, 5 = strongly agree) and focused on the following topics: the congruence of the process model with the actual project activities, the use of standards and training regarding processes.

Table II. Concentration of frequencies of the student-opinion poll (N = 3) on the process approach in the course of Program Engineering and Methodology

1 = I totally disagree; 2 = I disagree; 3 = I'm not sure; 4 = I agree; 5 = I totally agree.	**Likert scale**				
	1	2	3	4	5
Congruence					
The model is useful for guiding the activities of the process.	-	-	-	2	1
The level of detail in the model is adequate.	-	-	-	2	1
The activities described in the model correspond to those of the project.	-	-	1	2	-
The interactions of the model correspond to those that take place in the project.	-	-	-	1	2

It is easy to identify the subdivisions of the process.	-	-	1	-	2
Standards					
The patterns of the documents are suitable.	-	-	2	-	1
The quality attributes review lists are useful.	-	-	-	2	1
Training					
The class on theoretical processes is useful.	-	-	1	1	1
Communication with the tutor outside the class is important for clarifying uncertainties.	-	-	-	1	2

As we can see, encouraging results were obtained using the process approach in the course. Most scores are in the categories “I agree” and “I totally agree”. Naturally there are differences in the level of detail between the modeling process and the actual project activities. Caputo (1998) notes that the goal of process improvement is not to create a perfect process, but to generate processes that allow improvement in performance. In fact, Warboys *et al*. (1999) make a difference between generic process and tailored process. In the particular case of this study, we can consider that the model used has the key elements for achieving the goals of the process, but due to the nature of the software engineering course, is not adapted nor associated with individuals and machines to carry out the process, nor with the particular project that the students did. This could be one reason that some questions were rated “I’m not sure”.

It is necessary to pay greater attention to the documents generated during the analysis stage. The interview results indicate that there is a lack of clarity in the description of each of the parts of the document of requirements and specification. Also, there is an indication that it is possible to integrate the tool for generating the diagrams in UML language with the word processor so as to facilitate and automate the software specification document.

a. Congruence of the process model with the actual project activities. Students noted that the diagrams were useful as a guide to the work involved in their positions, because they let them visualize, quite simply, the activities they were doing; and allowed them to focus their efforts on a goal. The level of detail presented in the models was appropriate, since it made it easier for them to determine which activities to continue without specifying a particular technique for executing them. The models presented in the course met the stipulations Miers (1996) established, in the sense that a good definition of process must concentrate on its essence, reflecting the real world and managing that world’s complexity. In addition, is tailored to the indications of Ould (1995), who notes that a good description of process will be that which communicate the activities in detail to those who will carry them out, and will be precise enough to permit an assessment of compliance. Also, as expressed by Warboys *et al*. (1999), the RADS allow the description of complex behaviors in a highly legible manner, and achieve the appropriate capture of the organization’s rules.

b. Standards in the project. Participants indicated that the standards were a good guide for developing requirements and specification documents, and served as references in the revisions of these products. This result confirms what Humphrey (1989) has established, in the sense that standardization helps reduce the training problem. However, they suggested changing the order of some sections of the documents to improve the presentation, and requested an example written out in full to serve as a guide.

c. Training. The process theory class allowed students to orient themselves in the general context of software systems development, although they requested to be provided the process models for each of the positions in the company. They considered that the counseling sessions outside of class allowed them to express in a relaxed manner, the difficult aspects of the activities being undertaken, and helped them to focus their efforts on developing the software system.

The participants appreciated the importance of sharing information in a collaborative project, and identified the need for software processes to guide the team's practices and make the process visible. They realized the costs and benefits of teamwork, and became conscious that a software development project is far more than simply generating a code in a programming language.

VI. Conclusions

We have presented the problem of coordinating efforts among the participants of a software development project and the means used to mitigate it, from the perspective of processes in a software-engineering course. We have described, in a general way, the characteristics of a course in Program Engineering and Methodology taught in the CICESE, and the steps followed in the process-modeling project to investigate the interactions, activities, roles, documents and goals that are part of the software-development process used in the practical portion of the subject.

It was observed that the process-modeling techniques are effective tools for making the process visible, for identifying the particular problems of the software projects analyzed, and for identifying areas of the process that can be improved. In this study, the models isolated the issues related to the systems-analysis phase, which facilitated the updating of the topics in the theory session of the course and provided a blueprint for redesigning the processes belonging to that phase of the project.

The models described helped students gain a better understanding of their responsibilities in the process, and allowed them to focus their efforts on activities significant in developing the project; this gave them autonomy by reducing the number of interactions with the instructor. Preliminary results, using reengineered processes, suggest that models are a powerful guide for training project participants. Models facilitate collaboration between the various members of the group in identifying explicitly the types of interaction that exist at each stage of the development of a software system. It is not necessary for students to investigate what activities belong to their positions, nor with whom to interact during the process; they should only be concerned with understanding how to perform their tasks and tailor the software development project on which they are working.

VII. Future work

Work is currently being done on the key processes of Level 2 of the Capability Maturity Model of the SEI (Software Engineering Institute), in which are included: requirements management, project planning, project monitoring, quality assurance and configuration management (Paulk *et al*., 1995). These key processes will serve as a guide for the next course. Although the initial results indicate that process modeling aids students to learn about the activities they must carry out, especially when they have no experience in the area, it is necessary to test this model in graduate and undergraduate courses to corroborate the preliminary results previously described.

A second step, knowing the virtues of these models, is to extend them to a distributed development environment, where students are in different geographical locations. In this environment, the research will be focused on the interfaces which must exist in the process, in order for it to work properly; as well as recommending tools to support communication and coordination.

References

Arthur, L. J. (1992). *Improving software quality: insider's guide to TQM*. NY: John Wiley & Sons.

Ayer, S. (1992). *Documenting the software development process: A handbook of structured techniques*. NY: McGraw-Hill.

Bagert, D. J., Hilburn, T. B., Hislop, G., Lutz, M., McCracken, M., & Mengel, S. (1999). *Guidelines for software engineering education version 1.0*. (Technical Report CMU/SEI-99-TR-032 ESC-TR-99-002). Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.

Caputo, K. (1998). *CMM Implementation guide: Choreographing software process improvement*. Reading, MA: Addison Wesley-Longman.

Clark, B. K. (2000). Quantifying the effects of process improvement on effort. *IEEE Software*, *17* (6), 65-70.

Collofello, J. S., Kantipudi, M., & Kanko, M. A. (1994). Assessing the software process maturity of software engineering courses. *In Proceedings of the 25th SIGCSE Technical Symposium on Computer Science Education* (pp. 16-20). Phoenix, AR: ACM Press.

Curtis, B., Kellner, M. I., & Over, J. (1992). Process modeling. *Communications of the ACM*, *35* (9), 75-90.

Donaldson, S. E., & Seigel, S. G. (1997). *Cultivating successful software development: a practitioner's view*. Upper Saddle River, NJ: Prentice-Hall PTR.

Dorfman, M. & Thayer, R. H.. (1990). *Standards, guidelines, and examples on system and software requirements engineering*. Los Alamitos, CA: IEEE Computer Society Press.

Fairley, R. (1985). *Ingeniería de software*. (Trans. A. Sáchez & P. L. Flores-Suárez). Mexico: McGraw-Hill. (Original work published in 1985).

García Mireles, G. A. & Rodríguez Jacobo, J. (2000). *Manual de procedimientos para la elaboración del documentos de requerimientos en el desarrollo de software* (Technical Report CICESE CTCCT 2001, Serie Electrónica y Telecomunicaciones). Ensenada, B. C.: CICESE.

García Mireles, G. A. & Rodríguez Jacobo, J. (June, 2000). Propuesta para mejorar la enseñanza de la ingeniería del software basada en proyectos. Poster presented at *Congreso de Educación Abierta y a Distancia* (CEAD, 2000). Ensenada, B. C.:ANUIES-UABC-CICESE.

Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., & Paulk, M. (1997). Software quality and the capability maturity model. *Communications of the ACM*, *40* (6), 30-40.

Humphrey, W. S. (1989). *Managing the software process*. Reading, MA: Addison-Wesley.

Jaccheri, M. L. & Lago, P. (1997). Applying software process modeling and improvement in academic setting. In *Proceedings of the 10th* Conference on *Software Engineering Education and Training* (pp. 13-27). Virginia Beach, VA: IEEE Computer Society Press.

Kawalek, P. (1994). *Comments on the use of RADs in case studies*. Available at FTP: ftp://ftp.cs.man.ac.uk/pub/IPG/pk94.zip (June 10, 2000).

Kraut, R. L. & Streeter, L. A. (1995). Coordination in software development. *Communications of the ACM*, *38* (3), 63-81.

Licea, G., Rodríguez, J. & Favela, J. (1996). Evolución de procesos de desarrollo en la enseñanza de ingeniería de software. In *Proceedings of the 5th Congress*

Iberoamericano de Educación Superior en Computación (pp. 223-231). Mexico: Facultad de Ciencias, UNAM.

Maurer, F. & Kaiser, G. (1998). Software engineering in the internet age. *IEEE Internet Computing*, *2* (5), 22-24.

Mayr, H. (1997). Teaching software engineering by means of a "virtual enterprise". In *Proceedings of the 10th Conference on Software Engineering Education and Training*, (pp. 176-184). Virginia Beach, VA: IEEE Computer Society Press.

Miers, D. (1996). Use of tools and technology within a BPR initiative. In C. Coulson-Thomas (Ed.), *Process re-engineering: Myth and reality*, (pp.142-165). London: Kogan Page Limited.

Oktaba, H. & Ibargüengoitia, G. (1998). Software process modeled with objects: Static view. *Computación y Sistemas*, *1* (4), 228-238.

Ould, M. A. (1995). *Business processes: Modeling an analysis for re-engineering and improvement*. Chichester: John Wiley and Sons.

Paulk, M. C., Weber, C. V., Curtis, B., & Chrissis, M. B. (Eds.). (1995). *The capability maturity model: Guidelines for improving the software process*. Reading, MA: Addison-Wesley.

Pressman, R. (1993). *Ingeniería del software. Un enfoque práctico* (3ª. ed.) (Trans. C. Cervigon & L. Hernández Yañez). Madrid: McGraw-Hill. (Original work published in 1992).

Robillard, P. N. (1998). Measuring team activities in a process-oriented software engineering course. In *Proceedings of the 11th Conference on Software Engineering Education and Training*. Atlanta: IEEE Computer Society Press.

Rombach, H. D. (1990). Software specifications: A framework. In M. Dorfman y R. H. Thayer (Eds.), *Standards, guidelines, and examples on system and software requirements engineering* (pp.368-407). Los Alamitos, CA: IEEE Computer Society Press.

Sommerville, I. (1995). *Software engineering* (5ª ed.). Harlow: Addison-Wesley.

Tomayko, J. E. (1987). *Teaching a project-intensive introduction to software engineering* (Technical Report CMU/SEI-87-TR-20 ESD-TR-87-171). Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.

Upchurch, R. L. & Sims-Knight, J. E. (1997). Designing process-based software curriculum. In *Proceedings of the 10th Conference on Software Engineering Education and Training* (pp. 28-38). Virginia Beach, VA: IEEE Computer Society Press.

Upchurch, R. L. & Sims-Knight J. E. (1998). In support of student process improvement. In *Proceedings of the 11th Conference on Software Engineering Education and Training*. Atlanta: IEEE Computer Society Press.

Warboys, B., Kawalek, P., Robertson, I., & Greenwood, M. (1999). *Business information systems: A process approach*. London: McGraw-Hill.

Wastell, D. G., White, P., & Kawalek, P. (1994). A methodology for business process redesign: Experiences and issues. *Journal of Strategic Information Systems*, *3* (1), 23-40.

Translator: Lessie Evona York Weatherman

UABC Mexicali