Computación y Sistemas Vol. 3 No. 1 pp. 25 -37
© 1999, CIC - IPN. *ISSN 1405-5546* Impreso en México

Collaborative Software Development over the Internet: Tools and Experiences

Jesús Favela, Josefina Rodríguez, Guillermo Licea and **J. A. García**
Departamento de Ciencias de la Computación
Centro de Investigación Científica y de Educación Superior de Ensenada, B.C.
Km. 107 Carr. Tijuana-Ensenada, 22860, Ensenada, B.C., México
{favela, jacobo, glicea, antony}@cicese.mx

Article received on March 20, 1998; accepted on January 10, 1999

Abstract

The development of large-scale software systems requires the interaction of specialists with different areas of expertise. Communication and coordination among these specialists is of key importance to the success of a project. With the globalization of the software industry, these specialists are often distributed in several places complicating their coordination. This paper describes our experiences in the development of an Internet-based groupware environment to support software development. This environment, which integrates a set of tools that support several technical and administrative software development activities, was developed as the result of an evolutionary four-year process, where each year a different software development project was undertaken. We highlight the differences in coordination at different stages of this process arising from the use, or lack of use, of Internet-based collaborative tools.

Keywords:

Collaborative Software Development, CSCW, Internet.

1 Introduction

Large-scale software development is a collaborative activity that requires the interaction of specialists from different fields. These specialists need to communicate their decisions and coordinate their activities for the project to succeed. Project and configuration management techniques have been advanced for these purposes. Commercial CASE tools by in large, however, provide only limited support for explicit collaboration.

In the current software product development practice most technical information is communicated through specifications and diagrams. Coordination among project participants is achieved by using scheduling techniques and development processes that are known to all participants. These procedures, however, are increasingly being challenged by a competitive environment that demands better quality with shorter software development cycles, and teams that are often geographically distributed.

Several research efforts in the area of Computer Supported Collaborative Work (CSCW) have focused on supporting collaborative software development. Several systems have been proposed in the areas of requirement identification (Iochpe, 1995; Ramesh, 1992), configuration control (Grinter, 1995), coding (Brothers, 1990), project management (Ly, 1997), technical reviews (Gintel, 1993; Mashayekhi, 1994), process definition and enactment (Hutchens, 1997), among other.

Our research efforts focus on the design, development, and testing of collaborative tools to support the software development process. The collective use of these tools, while

facilitating the software development process, will allow project participants to record the project's memory, that is, the decisions made during the evolution of the project and the rationale behind them (Favela, 1997). We also investigate appropriate mechanisms for the retrieval of the group's memory to assist in product maintenance and new project development (Rodríguez, 1998).

As part of this effort we have developed the DISEL (Distributed Software Engineering Laboratory) environment which incorporates custom-made groupware tools that support several software development activities, such as, project planning and control; technical reviews; object-oriented analysis; configuration management; collaborative document production; information sharing; and Web-based software maintenance.

In this paper we describe the motivation behind the development of these tools, their functionality and our experience in their use for the development of medium-size software systems. These systems were developed in a Software Engineering course in which the graduate students and professional software developers that register in it work as a team. Section 2 discusses collaborative software development and the specific context that has motivated the development of these tools and their evolution. Section 3 describes the DISEL architecture and some of the specific tools that the DISEL environment supports. In Section 4, we discuss some of the lessons we have learned in the use of these collaborative tools. Finally, Section 5 presents the conclusions of this work and proposes future work.

2 Collaborative Software Development

This chapter summarizes some of the observations we recorded during the development of medium-size software systems and which lead to the design and development of groupware tools to address these problems (Licea, 1996).

The Capability Maturity Model (CMM) states that, as organizations reach higher levels of maturity, individual activities turn into team activities (Patnayakuni, 1995; Paul, 1993). Although the need for collaborative support in software processes has been recognized, how to efficiently support this collaboration is still a subject of debate.

To this end, several types of tools have been proposed to support collaboration, coordination and communication in software development. In one end of the spectrum there are traditional CASE tools, both commercial and research products, that are incorporating collaborative facilities, mostly in form of a workflow that uses a central repository of project information (product and process) to which all participants have restrained access. On the other hand there are collaborative applications aimed at supporting a specific software development task. These tools are often more sophisticated in terms of their collaborative features but have limited impact since they concentrate on specific tasks.

The approach followed in DISEL combines, to some extent, the previous two. Our aim is to create a flexible environment that incorporates a variety of loosely integrated tools that take advantage of shared resources and which could be more or less valuable depending on the nature of the project being undertaken (group size, group location, etc.). Rather than defining a complete and complex environment from the start, we have followed an evolutionary approach. In this approach we first identify the more pressing communication and coordination problems within a software development project, design the processes and tools that can better address this problems, implement the tools and test them in a new project in which we identify new collaboration problems that results in the modification of these tools or the development of new ones.

2.1 The Software Development Projects Analyzed

We have followed the development of 4 medium-sized software systems in the last four years. During the first three years, the systems were developed in a period of 12 weeks by 14 to 20 member groups, which combined graduate students and professionals from regional software development companies (Table 1). A fourth course lasted 32 weeks and included students from institutions in two different countries: MIT in the USA and CICESE in Mexico. In all of these projects each participant assumes a role in the project, one of them acts as the Project Manager, another one is in charge of quality control, two or three of them design the system, etc. The members of the team worked at up to four different locations in up to two different cities (projects 2 and 4). All participants meet once a week to report advances and coordinate activities. In addition to these weekly meetings they communicated using different facilities such as e-mail, telephone and fax.

Project	Number of participants	Location of group members	CSCW support
CRAB	16	Co-located	e-mail
EFYCAZ	14	Distributed	LotusNotes
SICC	20	Distributed	Kiliwa WWW environment
DISEL	18	Fully distributed	DISEL WWW environment

Table 1: Software projects studied and supported by CSCW tools.

2.2 Communication and Coordination Issues Raised in the Projects

In the first project computer-mediated communication was limited to e-mail. For the second project we designed and implemented LotusNotes databases that were used in the last five weeks of the project to support project and configuration management activities. In the third project, participants used the Kiliwa collaborative support environment which integrates a suite of groupware tools on the Internet. The development of Kiliwa was motivated by our experiences and observations of the first two projects. The experiences of the third project were used to enhance Kiliwa into what we call the DISEL (Distributed Software Engineering Laboratory) Web-based software development environment. In the reminder of this section we describe our experiences with the first two projects. Those from the last project are presented in Section 4.

2.2.1 The CRAB Project

The CRAB system was developed under request of a local library to provide distributed access to its catalog. The project reported several delays and coordination problems. These deficiencies were mostly attributed to poor group integration (Licea, 1996). Nevertheless, we identified a couple of issues that could lead to improved group performance with appropriate support tools. These were project control and configuration management as explained ahead. It is not surprising that these activities correspond to two of the Key Processes Areas to which organizations should focus their efforts in order to achieve Level 2 maturity according to the CMM, namely Software Project Tracking and Oversight and Software Configuration Management (Paul, 1993).

Given the tight development schedule (the project had to be completed, from analysis to testing and delivery in 12 weeks), project participants had to report frequently on the status of the tasks assigned to them. These reports were handed out to the Project Manager on the weekly meetings. Progress reports were often late complicating the Project Manager's task of estimating delays and re-assigning work. Furthermore, project participants lacked awareness of overall project status.

Change control was also stressed by the tight development schedule. Once change requests began to arrive with frequency by the late stages of design, the Configuration Manager found difficulties in resolving these requests in a timely manner. The delays in resolving and communicating the decisions of the Change Control Board to the rest of the group added to the delays and confusion by project participants.

2.2.2 The EFYCAZ Project

The EFICAZ system was developed for a professional service organization to keep control of their personnel assignments and work record. For this project we developed a couple of LotusNotes databases to support the project tracking and configuration management. The first database allowed participants to submit their weekly reports, review their individual progress as well as the tasks assigned to them. Overall project status was still communicated during the weekly meetings. The second database allowed participants to place a Change Request Order from their terminals. The system helped the Configuration Manager to keep track of change requests and their status. The decision of whether to accept or reject a given change request were resolved by voting through electronic means and occasional meetings.

These databases, evaluated during the second project, were found to be useful, although not completely convenient. Chief among their limitations was the fact that we had only a limited number of computers equipped with LotusNotes clients, and some of the participants found it difficult to access them to send their reports or request a change. This motivated us to port these tools to the World Wide Web and enhance them to address some of their other limitations.

The second project was considered successful, although we wouldn't argue that this was directly caused by the new tools developed to enhance computer-mediated communication and coordination. However, the success was largely based on improved team harmony (Licea, 1996), defined as the facility with which project participants coordinate their work and communicate about design issues with each other (Souder, 1988). In addition, we identified the distribution and review of analysis and design diagrams as an area in which CSCW tools could help improve the next project. This is further explained next.

Most technical communication among group members referred to documents produced within the group, specially the analysis and design diagrams that were developed using the object-oriented methodology Object Modeling Technique (OMT) (Rumbaugh, 1991). These diagrams were reviewed in separate technical review meetings. They were modified following the recommendations from these meetings and frozen afterwards as versions 1.0. Subsequent revisions to these diagrams had to follow a change control procedure administered by the Configuration Manager. Change requests were evaluated by a committee. If the suggested change was accepted this decision was informed to all the participants affected by it and a change order was sent to the person responsible for implementing it.

Most of the changes to the analysis and design diagrams were originated when other members of the team used them down the development process. For instance, most of the change requests to the design diagrams were originated by the programmers while in the process of implementing the

design. The programmers' careful review and discussion of the design was complicated by the fact that the programming group made of four people was geographically distributed. A typical scenario was as follows: One of the programmers would find what he thinks is a mistake in the design, he will discuss this to other members of the programming group by e-mail, or more frequently, by telephone. In the former case, the two programmers will make extensive references to the design diagrams, which they had at hand, during the conversation. Once an agreement is reached to report the defect one of them fills a Change Request Form that is given to the Configuration Manager to be discussed during the next configuration committee meeting. Although at the end of the project this last part of the process was supported by a workflow system implemented in LotusNotes, the programmers' work was often suspended until a resolution to the change request was reached and the design diagrams were modified. In some instances, the changes to the diagrams originated new change requests.

As part of the development process revised versions of the design diagrams have to be frequently distributed to the programmers. This was done using fax or snail-mail for those programmers that were not co-located with the design team. Considering that the project had a very tight schedule, short delays in the reception of the design originated communication problems between the programming team that worked with different versions of the design.

The problems just described motivated us to find a better mechanism to distribute the analysis and design models and to facilitate the interpretation and discussion of the information contained in them. Our first alternative was to import the diagrams into LotusNotes which could give us the additional advantage of supporting argumentation threads of the analysis and design diagrams, a limited form of technical review. At the end, we opted for a different solution based on the automatic generation of HTML documents from the CASE tool used to assist during analysis and design. This was implemented in the ICARO system.

ICARO (Licea *et al.*, 1996b) extends a commercial CASE tool to allow for asynchronous distributed collaboration over the WWW. The extended tool can be used by project participants to keep them updated with changes in the analysis and design diagrams, to inquire about technical issues of the project that is being co-developed, and to argue the merits of a particular design decision.

The commercial tool, which supports the creation of object oriented analysis and design models, is extended to automatically generate HTML versions of these documents. The HTML files can be consulted by all team members, even if they work in different places. The HTML tiles include calls to Java applets that allow the reviewers to ask questions about the models, argue about the merits of decisions made, or suggest changes to be made. These arguments are stored in argumentation threads that can he consulted using a WWW browser.

2.2.3 The SICC Project

SICC is a web-based information system developed for the local Chamber of Commerce. In this project most of the documentation produced (design diagrams, test cases and results, project plan, etc.) were stored as HTML documents accessible to all team members. There was, however, poor document integrity and some redundant information. The document specialist, responsible for the web-repository, had trouble maintaining the integrity of the site which evolved quite fast as the project progressed.

Communication was enhanced with respect to the previous project, since all participants could rapidly find updated documents from any computer using a Web browser. Some documents, however, were not reported to the document specialists and thus were not reachable from the site's main page. This was often the case for preliminary versions that were being co-developed in subgroups. The effort required to maintain the site was considerable, but the benefit of the repository goes beyond the project's development effort as it should facilitate its maintenance and be a valuable reference for future projects.

During this project, the ICARO tool was no longer used. The reason was that the CASE tool used to create the analysis and design diagrams was substituted by a more powerful one with support to directly export the diagrams to HTML. Thus, we started development of the SCART system, to support the technical review of all kinds of software development documents. The development of this tool coincided with the need for a more formal review processes that included inspections and walkthroughs and not only the informal argumentation of documents supported by ICARO. SCART is explained in greater detail in the next section.

As a result of this experience we set the following priorities: a single access point to all the tools and data to the user, and the integration of data used by all the tools. The resulting environment is presented in the next section.

3 The DISEL Environment

In this section we describe the elements that constitute the DISEL environment. Some of these elements were purposely developed for use with DISEL, and some others were developed previously and integrated into DISEL.

The DISEL environment has a multilayer architecture that allows DISEL applications to access a data base with all the documentation (text documents, diagrams, images, schedules,

etc.) produced during the development of a software project. DISEL applications can share these documents and use them in different phases of the development process. Figure 1 shows the architecture of DISEL.

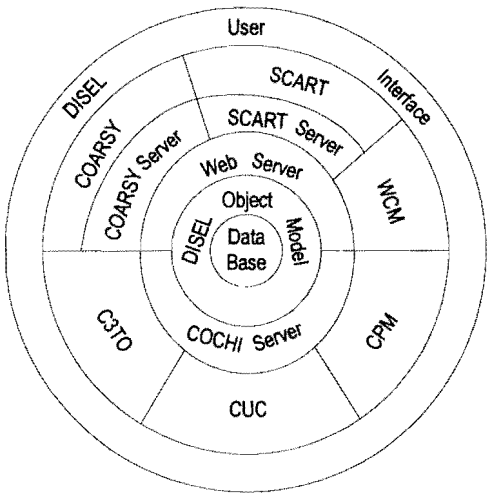


Figure 1: The architecture of DISEL.

In the deepest layer is the Data Base which contains the documentation produced during the development of a particular software project. These documents can be accessed by all DISEL applications through the DISEL Object Model, which provides a mechanism for different applications to share the documents generated by others.

The DISEL Object Model is included in the second layer and provides a product and process model of software development. The DISEL Object Model helps the Web and COCHI servers store and retrieve documents or records from the Data Base to or from the Data Repository. The DISEL Object Model presents a higher level of abstraction to the applications. In this way, for instance, the group could use a project management tool to schedule a technical review meeting and the tool used to support this meeting will be aware of it. The DISEL Object Model is partially implemented to support data connectivity between several of the tools in DISEL.

On top of the DISEL Object Model layer, there is a third layer that contains the different servers used by DISEL. There are two servers: A web server that supports asynchronous applications that operate through the Web and the COCHI server, described in the next section, which provides a basic collaboration infrastructure for synchronous applications. The first group of applications might also require dedicated servers as is currently the case of the COARSY collaborative document editing and review system and SCART, used to support technical review meetings.

The next layer corresponds to the DISEL applications: COARSY, WCM and SCART that are web-based and C^3TO, CUC, and CPM that were implemented as Java applications based on COCHI's platform. All of these applications can be invoked by the DISEL user interface (essentially a web-page with links to applications and data), which forms the last layer of the DISEL architecture. There are therefore two levels of integration between all applications. At one end, is the DISEL Object Model, which provides a common interface to access data produced by all the applications. At the other end, is the DISEL user interface, a single access point to all DISEL applications. A new application could be loosely integrated to the system by just adding a link in the DISEL user interface or could achieve a higher level of integration by sharing with other applications through the DISEL Object Model.

In the remainder of this section we present the different tools already integrated in DISEL, starting with a brief description of COCHI. These tools are grouped in three main areas: Requirement Analysis, Project Management, and Control, the last of which includes Quality Control and Configuration Management.

3.1 Collaborative Software Development Tools Based on COCHI's Platform

COCHI (Collaborative Objects for Communication and Human Interaction) (Licea, 1998) is a platform that supports the development of collaborative applications based on a reusability technique called pattern systems (Buschmann, 1996).

COCHI, as a pattern system, contains software patterns that help in the design and implementation of a collaborative application. It includes an architectural pattern that defines the general structure of a client-server based collaborative application, a set of design patterns that describe the subsystems that form the architectural pattern and two Java class frameworks that implement the design patterns and support the rapid development of collaborative applications.

The first element of the COCHI pattern system is a layered architecture composed of several subsystems. The architecture itself is based on a design pattern (Client-Server design pattern) that allows participant's applications to communicate with each other through a server using a predefined protocol. Figure 2 shows the architecture of COCHI.

The *server* is composed of three layers. The Client-Server design pattern includes all the necessary elements to allow communication between a server and applications connected to it. The collaboration layer of the server includes five subsystems that extend the Client-Server design pattern included in the server layer. The configuration layer allows participants to configure the characteristics of the session.

The *participant* includes four layers. The client layer includes the Client-Server design pattern. The collaboration layer of a session participant has the same five subsystems included in the collaboration layer of the server, so each

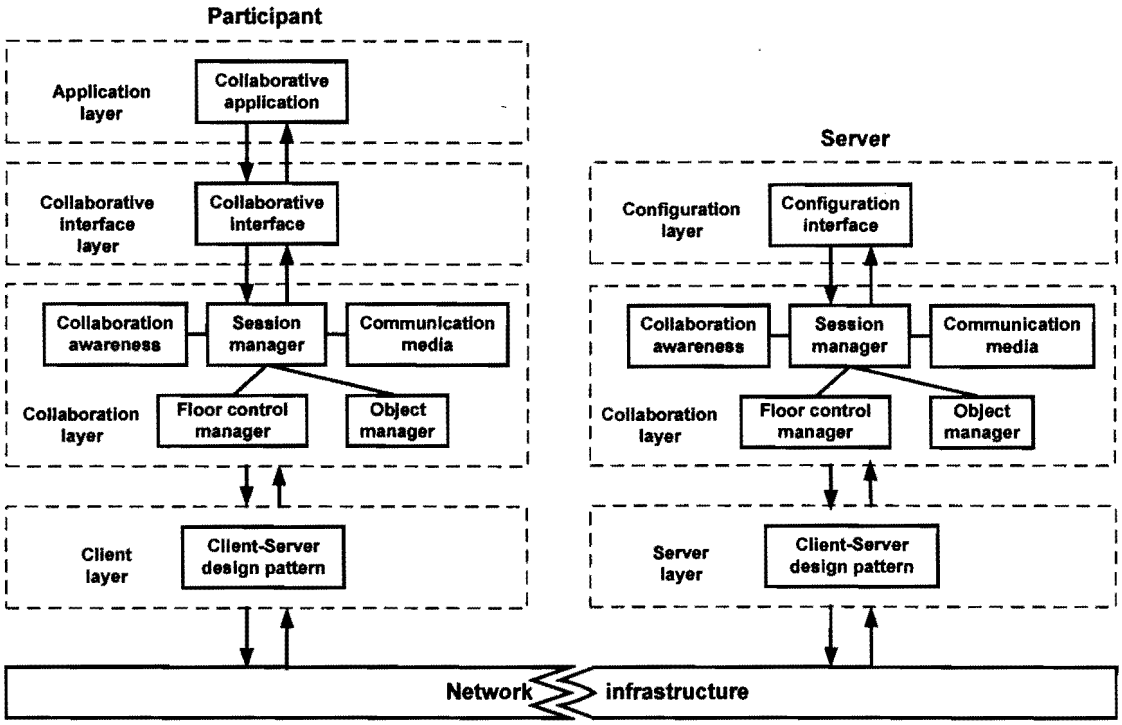


Figure 2: Architecture of COCHI.

subsystem of the participant interacts with the corresponding subsystem of the server. The collaborative interface layer represents the interface between the developer and the collaboration layer, its features and services. This layer includes the necessary elements for developing groupware applications based on the COCHI platform. The application layer is the groupware application itself.

The Java class frameworks of the COCHI pattern system contain the implementation of all the classes included in the design patterns. These frameworks have been used for the development of several small and medium-size collaborative applications like some of the ones included in the DISEL architecture.

3.2 Requirement Analysis Tools

Requirement analysis is one of the most important steps of the software development process. It is the first technical task and sets the basis for the process. Some of the most common reasons of project failure have to do with poor requirement analysis: failure to identify the client's real needs, feature explosion, failure to control the requirements, etc. This is a collaborative process in which at least two individuals participate, the client and the analyst. The recognition of this fact has given impulse in recent years to area known as Participatory Design (Muller, 1993).

DISEL incorporates two collaborative synchronous tools to support this process: The CUC (Collaborative Use Case) diagramming tool and C3TO to facilitate Object Oriented Analysis using the Class- Responsibility-Collaborators technique.

3.2.1 Drawing Use Case Diagrams with CUC

Use Case diagrams are used to describe the typical scenarios of use in an information processing system (Jacobson, 1992). A use case diagram shows a set of external actors and their connections (or relations) to the use cases that the system provides. These diagrams help to obtain the requirements specification of a software system during the object-oriented analysis phase and have also been found useful in the scheduling and management of projects since they can be used to divide and identify the work to be done.

The Collaborative Use Cases drawing tool (CUC) is a collaborative application that can be used to draw use case diagrams in a synchronous groupware session. Figure 3 shows the main windows of this application that was implemented with the class frameworks of the COCHI platform. COCHI's frameworks allow developers to include features such as collaboration awareness, communication media, floor control mechanism, session management, that are common to most synchronous groupware.

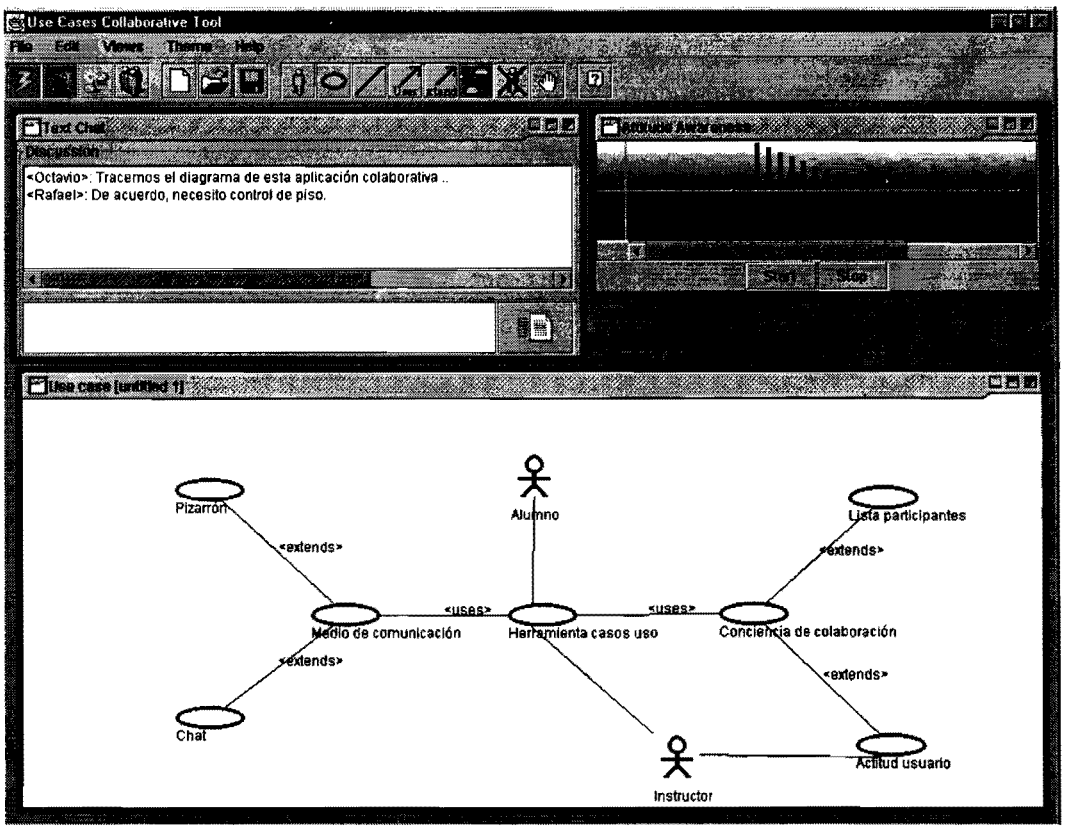


Figure 3: Drawing Use Case diagrams using CUC.

3.2.2 Collaborative CRC Case Tool

C³TO is a synchronous collaborative tools written in Java which supports the analysis of scenarios in the object-oriented analysis of a system using the CRC (Class-Responsibility-Collaborators) technique (Beck, 89). This technique proposes the use of cards which represent classes. These cards are filled with the following information: the classes' name, its responsibilities, and its interactions. These cards are created using scenarios, taken from the requirements of the system, which model their behavior. Each card is divided in three sections. At the top-left of the card is its name, to the left is the list of responsibilities and in the right are the collaborators, or other classes with which this card is related.

The C³TO system allows for the interaction of members of the analysis team including the analyst and a domain expert. It allows them to work at the same time in the analysis of scenarios while filling the cards. The tool provides a WYSIWIS (What You See is What I See) interface for card edition in which all participants see the same information. Only one person can edit the cards at a time. The floor control has to be explicitly requested and granted. Users communicate by written messages or using an audio or videoconferencing application external to C³TO. This tool was developed using the COCHI pattern system.

3.3 Project Management

Three tools have been developed to support project management: CPM, a synchronous collaborative tool for project planning, ProCon, a workflow system to keep control of individual's progress and overall project status, and COARSY, an application for the collaborative edition and review of HTML documents.

3.3.1 Collaborative Project Management (CPM)

CPM is a synchronous collaborative tool for project planning. It allows the project manager to specify and schedule project activities. These activities could be displayed using Gantt chart and they can be analyzed using the Critical Path Method (CPM). In contrast with other project planning tools, CPM can be used simultaneously by two or more users who can collaborate in the definition of activities, the estimation of their duration, analysis of delays and rescheduling. In a typical scenario, the project manager might start a session with one of the participants to discuss the status of an activity for which

he is responsible. Through discussion and negotiation and by analyzing the effects on the whole project, they might decide to give additional time for this activity to be finished and make all appropriate adjustments to the project plan. CPM was implemented using the COCHI pattern system. Figure 4 shows the interface of the tool, including the list of activities, a Gantt chart, a list of shared document with progress reports and textual communication with collaboration awareness as pictures of all participants in the session.

3.3.2 Project Control (ProCon)

ProCon was developed as a set of CGI-scripts accessed via HTML forms to help the project administrator with this tasks such as sending project status updates to all members and sending weekly lists of assigned activities to individual members or groups.

Every week, each member of the project accesses a report form and describes all activities performed during that week, and reports progress (percentage of completion and reasons for delays) on those activities explicitly assigned to him. Once submitted these reports are sent to the project manager, and optionally, to all project members who asked to receive such reports. ProCon also provides HTML forms meant to be filled by the project administrator to send tasks assignments to individual project members or sub-groups.

3.3.3 Project Documentation (COARSY)

A considerable amount of a software developer's time is spent reading or writing documents. The creation of these documents is, in most cases, a collaborative activity. The most common approach might be for one individual to prepare a draft to be handed out for review to others. COARSY is a collaborative system that facilitates the edition and review of HTML documents (Ruiz, 1998). A document that is to be co-authored or reviewed is registered in the system. Each participant is assigned a role (co-author or reviewer). A co-author can modify the text and add contributions that can be of several types (change, comment and question). A reviewer cannot alter the text but he can make contributions. The contributions can be made at different levels of granularity (document, section or paragraph), they can anonymous or identified. Figure 5 shows the interface of the COARSY system. The main window is divided in two frames, one displays the document, and the other one the contributions.

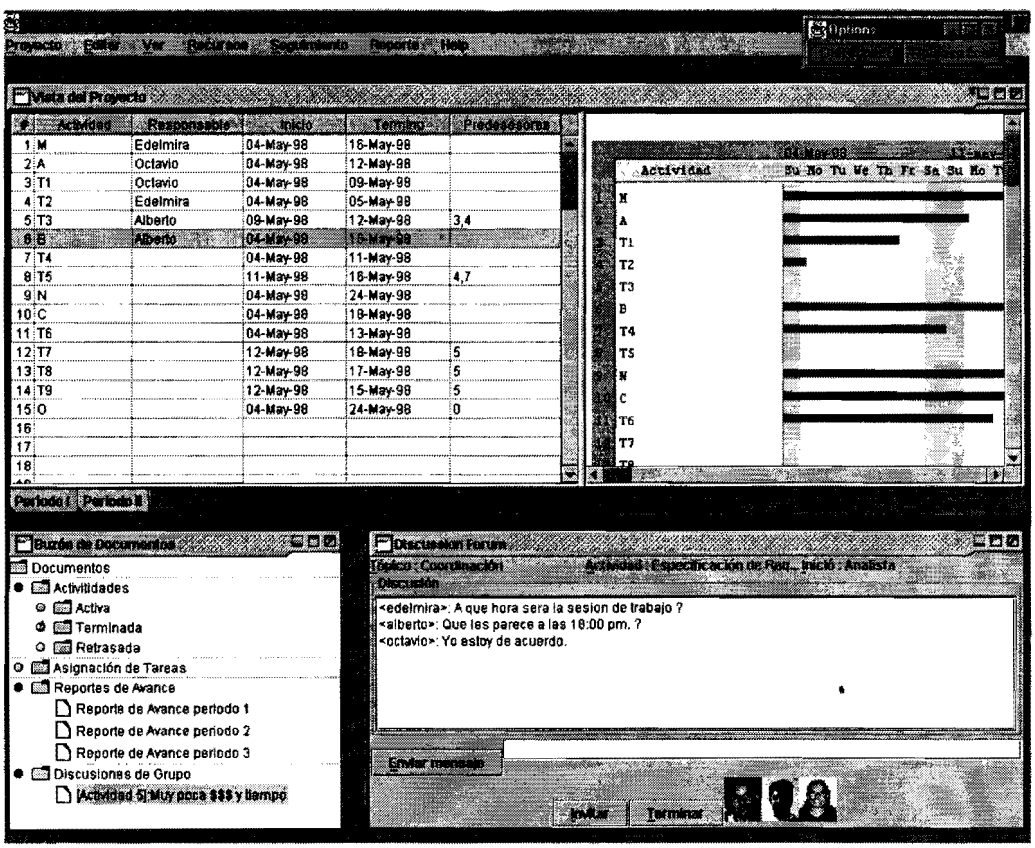


Figure 4: Project planning using the CPM tool.

By selecting one element of the document, a paragraph, for instance, all contributions made to this element are highlighted and displayed in the bottom window, this greatly simplifies making corrections to the main document. The example shown in Figure 5 corresponds to a document sent by a project manager proposing changes to the original schedule, the changes are discussed by all participants using COARSY.

3.4 Quality Control and Configuration Management

The third group of DISEL applications helps control de development of the project and correspond to the areas of quality control and configuration management. The first tool, named SCART, supports distributed technical review meeting, while WCM is a web-based change control system.

3.4.1 Managing Change in Software Development (WCM)

Configuration management combines tools and techniques to control the software development process; it controls different versions of documents generated through the project's life cycle and keeps control over modules that have been coded in order to avoid update and simultaneous maintenance problems.

A software tool named WCM (Web Configuration Manager) was developed for the DISEL system. WCM is a workflow automation system that allows any project participant to request a change. He does so by filling a Web form that is sent to the Configuration Manager.

Every time a change request is made, a web page with information related to the request is automatically generated and a message is sent to the Configuration Manager indicating where the generated page is. After reviewing the change request he can inform the members of the change control board to review the change request and express their opinion as to whether it should be accepted or not. If the change is accepted, change orders are e-mailed to those involved in such change, if it is not accepted e-mail is sent to the person(s) that requested the change informing him of the decision. A management module allows the Configuration Manager to keep track of all requests made and their status.

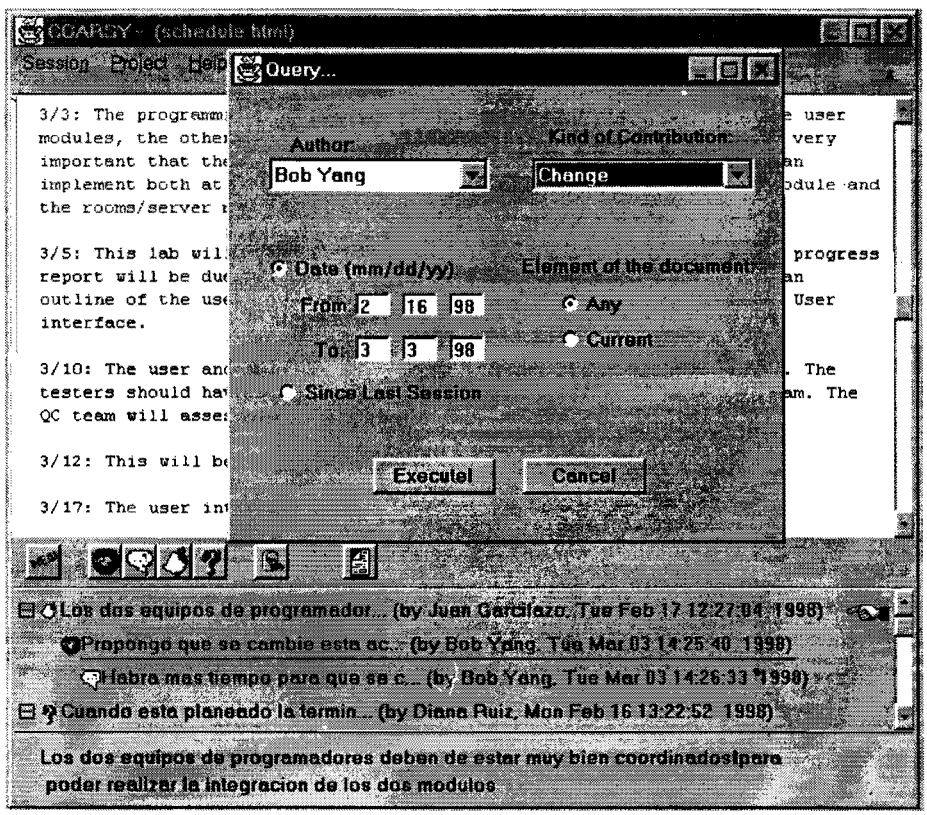


Figure 5: COARSY permits the co-authoring and review of HTML documents.

3.4.2 Technical Review Meetings (SCART)

SCART supports the review of technical documents by a local and/or distributed group of reviewers, in a way similar to traditional review meetings (Fagan, 1976). The review meetings supported are of two types: inspection and walkthrough. The main idea is to reduce the stress of the participants in the meeting, and reduce the costs of travels when the reviewers are geographically distributed.

To begin the technical review meeting the document to be reviewed is registered in the system along with all participants. The reviewers are electronically notified of their participation and the location where they can find the document (see Figure 6). In an inspection, first an asynchronous review takes place, in which each participant uses SCART to review the documents and register the defects found in it. In a second phase the meeting takes place. Participants to the meeting can be co-located or distributed and they use SCART to navigate through the document as each reviewer describes the defects found in it. In this phase, the web-browsers are synchronized by SCART, thus, all participant see the same information controlled by the one control of the floor.

4 Lessons Learned with the Use of DISEL

Before we present our observations in the use of DISEL, we first describe the characteristics of the project in which this environment was used which differed in some important aspects from the previous projects.

In contrast with the previous projects described in Section 2, the DISEL project lasted 32 rather than 12 weeks and had 18 students from institutions in two different countries: MIT in the USA and CICESE in Mexico. This also added the difficulty that half the group had to communicate using their non-native language (English).

The last important difference with the previous project had to do with the nature of the problem to be addressed. In previous projects the team had to obtain the requirements from a client. This trained them in requirement gathering techniques such as interviews, surveys, etc. For this project the requirements were very general "develop a tool to support collaborative distributed design" and the group had to come up with an innovative product. This forced the group to communicate more during the first phase of the project but also raised the tension between the participants, in particular those in charge of requirement analysis and specification.

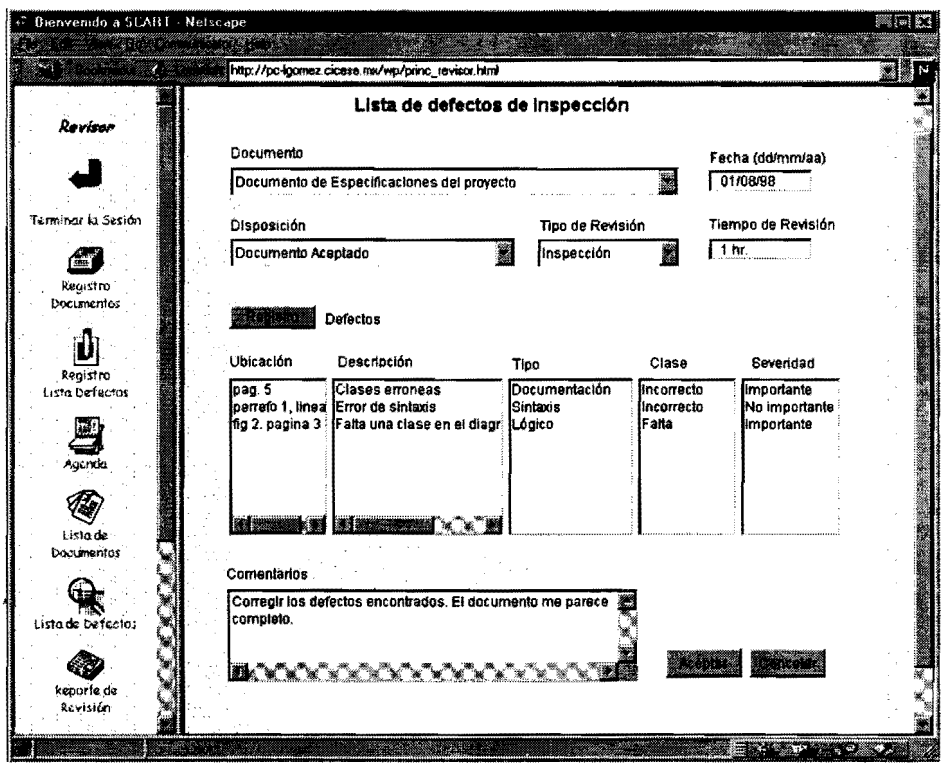


Figure 6: List of defects found in the inspection of a document using SCART.

Given the distributed nature of the project, all formal meetings with all members of group were done using videoconferencing equipment. Given the additional technical difficulties special emphasis was placed on elaborating agendas for the meetings and controlling the time planned for each item in the agenda.

Every month a survey was conducted among all participants to measure a number of issues ranging from aspects such as team harmony and the use of tools. We also analyzed the electronic messages sent to all participants and the issues discussed in the meetings.

Of particular interest was the participant's perception and ability to communicate with other group members and follow the progress of the team. In both cases the group's perception was that the project started with poor communication and coordination, this was raised as the project progressed and hit a new low towards the end of the project. This was in part due to the fact that most participants were at the end concentrated on their own work, which was more clearly defined and there was less need for close collaboration. Interestingly, their overall satisfaction with the experience followed the opposite trend, started being good, it went down as they found difficulties coordinating their work and it went up again in the last two months of the project as they learned to deal with the issues presented by the project.

Several of the tools provided by the DISEL environment were used frequently and were considered valuable. In particular, having a repository with all project information was rated high. They could all access the different versions of the documents being produced to review them and/or used them for their own work. Diagram and document sharing through the WWW solved the distance problem, accelerated the analysis and design stages and allowed programmers to begin their work while the design was being corrected.

This configuration management tool helped the group to have a better and quicker response to changes in documents, diagrams and code, minimizing the idle time and accelerating the development process. It was mostly used during the design phase.

Technical review meeting were partially supported by SCART. The system was used mostly in its synchronous version since most reviews were walkthroughs rather than inspections. In this case, one SCART client was used in each side with a projection system for all participants to follow the issues being discussed during the review.

The requirement analysis tools (CUC and C^3TO) were not used much. The analysts preferred to work alone and then distribute the document to their college for its review. In this phase a requirement tracking system could have been more useful.

One of our findings is that being the group distributed there was a very high overhead to contact a college in the remote location. Even if only simple information was required an e-mail had to be sent, or even worst, a meeting scheduled. This has led us to design an environment to support informal opportunistic interactions among project participants when they access project information. The system operates as follows: when user request a web page from the repository his userID and current URL is sent to an awareness server that has information of all current users' location. Every time a user moves to a different page, this information is updated to all other users and displayed in a separate window. This way all users are aware of the information other project participants are browsing at a particular moment in time. By selecting a user from the awareness window a user can request to initiate textual communication with that user (eventually audio and video). If the other user accepts the request they can start communicating. The awareness architecture is similar to the one proposed in (Palfreyman, 1996), but it is only implemented in the web server where the project repository is contained and allows collaboration among users. We expect to test this system in a new project shortly.

5 Conclusions

Communication and coordination in software development projects has been a subject of research for several years. Most research in this area concentrates on observing work in-progress (Button, 1996; Kraut, 1995; Potts, 1996;Walz, 1993). These studies, however, do not evaluate the use of groupware, but mostly concentrate on other coordination techniques and artifacts, such as meetings, project and configuration management techniques, etc.

On the other hand, it's been widely recognized that groupware systems need to be evaluated while in use, given the complexity of human collaboration (Grudin, 1989). However, most of the groupware tools developed in the last few years to support the software processes have not been evaluated at all, and if so, only in the limited extent of the specific task that they support (Atwood, 1995). The research proposed in this paper tries to reconcile these two extremes (evaluate team work without groupware and develop groupware which is not evaluated).

To this end, we have followed a project-based Software Engineering graduate course over the last four years in which we have developed medium-sized projects with the purpose of understanding the collaborative software development processes and how groupware tools can enhance these processes. Based on this, we have implemented and tested several groupware tools which have lead to the DISEL software development environment. We plan to enhance DISEL in the directions discussed in previous sections.

Some concluding remarks based on the experiences discussed are the following:

- The tools developed have been proposed to attend very specific concerns that were raised from the experience of previous projects.
- All these tools support the development process being followed in the project and which aims at achieving a level 2 maturity according to the CMM model (Paul, 1993).
- The Web facilitates the loosely integration of these tools. By simply adding appropriate links one can easily move from one tool to another and to the project's documentation as required.
- The Web raises the expectations of the users in the sense that they expect up-to-the-minute information of the project and this can lead them to think that it is not necessary to consult other sources of information and forces the group to continuously update the web server.

Based on our experience, the DISEL environment is a step in the right direction of supporting the whole software development process. In a similar way in which the CMM recommends software organizations to progress gradually to higher levels of maturity by focusing first on the most pressing issues, we have followed an evolutionary approach in which new tools introduced respond to the experience of previous projects.

There are, however, important limitations in the work presented here. As described above, the tools adhere to a specific software development process being enforced on the group. Most of these tools might not be useful to a group developing by prototyping or following a more informal development process. Also, we have been both responsible for the projects as well as for implementing and testing the tools described. In this regard we cannot be totally objective and by asking the group to use a particular tool we might be interfering in the experiment itself. Finally, there is a considerable effort required to set up the DISEL repository before the start of each project as well as during its maintenance.

The DISEL environment is just the beginning in the way for a more mature environment, such as Baenscht's vision of a Global Software Highway (Baensehi *et al.*, 1995), but our experience tells us that the web adds new capabilities (global distribution. browser-based access) to rather well established software engineering methods.

In the near future we will focus on supporting the collaborative modeling of software processes and not only their enactment (Patnayakuni, 1995; Araujo, 1999). This focus corresponds to the requirement for organizations to define the processes they follow in order to reach the third maturity level of the CMM.

Acknowledgments

The authors wish to acknowledge Octavio Garcia, Lidia Gomez, Rene Navarro, Diana Ruiz, and Marcela Rodriguez who contributed to the development of the DISEL environment, as well as Prof. Feniosky Pena-Mora from MIT. This work was partially supported by CONACYT grant No. C024-A and graduate scholarship No.117161 provided to the third author.

References

Araujo, R., Borges, M., Dias, M., Favela, J., Rodriguez, J., "Supporting Software Processes with Groupware." In Proc. of the Encuentro Nacional de Computacion ENC'99. Pachuca, Mexico. Sept. 1999.

Atwood, M. E., Burns B., Gairing, D., Girgensohn, A., Lee, A., Turner, S. Alteras-Webb, S., and **Zimmermann, B.** "Facilitating Communication in Software Development." In Proceedings DIS '95: Symposium on Design Interactive Systems, Ann Arbor, MI, August 23-25. New York, ACM Press. 1995. pp. 65-73

Baenscht, M., Molter, G., Sturn, P. "Booster: A WWW-based Prototype of the Global Software Highway", Proc. 2nd International Workshop on Services in Distributed Networked Environments SDNE'95, Whistler, Canada; IEEE Computer Society Press, June 1995.

Beck, K. and **Cunningham, W.A.** "Laboratory for Teaching Object-Oriented Thinking" Proc. of the OOPSLA '89 Conference. Oct. 1989, New Orleans, LA, pp. 1-6.

Brothers, L., Sembugamoorthy, V. and **M. Muller.** "ICICLE: Groupware for Code Inspection." CSCW'90, 1990, pp. 169-181.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. and **Stal, M.** A system of patterns: Pattern-oriented software architecture. John Wiley & Sons. 1996.

Button, G., and **Sharrock, W.**, "Project Work: The organization of collaborative design and development in software engineering." Computer Supported Cooperative Work. Vol. 5, No. 4. 1996, pp. 369-386.

Fagan, M. "Design and code inspections to reduce errors in program development." IBM Systems Journal, Vol. 15, No. 3, pp. 182-211, 1976.

Favela, J. "Capture and Dissemination of Specialized Knowledge in Network Organizations." *Journal of Organizational Computing and Electronic Commerce*, Vol. 7, 1997, pp. 201-226.

Gintel, J. E., Arnold, J., M. Houde, Kruszelnicki, J., McKenny, R., Memmi, G. "Scrutiny: A Collaborative Inspection and Review System." *In Proc. of the 4th European Software Engineering Conference*. Garwisch-Partnkrichen, Germany. September 1993.

Grinter, R. "Using a Configuration Management Tool to Coordinate Software Development," *In Proc. of the ACM Conf. on Organizational Computing Systems*: COOCS'95,

Milpitas, CA, August 13-16, 1995, pp. 168-177.

Gruding, J. "Why groupware applications fail: Problems in design and evaluation." *Office: Technology and People*, Vol. 4, No. 3. 1989. pp. 245-264

Hutchens, K., Oudshoorn, M., Maciunas, K., "Web-Based Software Engineering Process Management." *In Proc. of HICSS'97*. Maui, Hawaii, January, 1997, pp 676-685

Iochpe, C. "Improving Requirements Analysis Through CSCW," *In Proc. of the First International Workshop on Groupware CRIWG'95*, Lisbon, Portugal, Sept. 1995, pp. 29-37

Jacobson, I., Christerson, M., Jonsson, P. and **Overgaard, G.** Object-Oriented Software Engineering: *A Use Case Driven Approach. Addison-Wesley. 1992.*

Kraut, R., Streeter, L. "Coordination in Software Development." *Comm. ACM*, Vol. 38, No. 3. 1995. pp. 69-81.

Licea, G., Favela, J., "ICARO: A Web-Based Environment for Collaborative Software Development." *2nd. International Workshop. on Groupware, CRIWG'96*. Puerto Varas, Chile, Sept 1996, pp. 23-30

Licea, G and Favela, J. "Design patterns for the development of synchronous collaborative systems." *In Proc. of the Taller Internacional de Tecnología de Software, Simposium Internacional de Computación CIC-98*. 1998. pp. 51-61.

Licea, G., Rodríguez, J., Favela, J. "Evolution of software development processes in distributed heterogeneous and loosely coupled groups. (In Spanish). *Proc. of the 5th Latinoamerican Congress on Higher Education in Computing, EDUC'96*. Sept. 1996, pp 223-231.

Ly, E. "Distributed Java Applets for Project Management on the Web." *IEEE Internet Computing*. Vol. 1, No. 3. 1997. pp 21-26.

Mashayekhi, V., Drake, J., Tsai, W., Riedl, J. "Distributed Collaborative Asynchronous Inspection of Software." *IEEE Software*. Sept. 1993. Pp. 66-75

Muller, M., Kuhn, S. "Participatory design." *Comm. of the ACM*, Vol. 36, No. 6, 1993. pp. 24-28.

Patnayakuni, R., Patnayakuni, N. "A Socio-Technical Approach to CASE and the Software Development Process." *In Proc. of the 1st American Conference on Information Systems*, Pittsburgh, Pennsylvania, August/1995, pp. 6-8.

Paul, M. C., Curtis, B., Chrissis, M. B., Weber, C.V. "Capability Maturity Model for Software v.1.1." *Technical Report CMU/SEI-93-TR-24*, Carnigie Mellon University, Software Engineering Institute, February, 1993.

Palfreyman, K. and **Rodden, T.**, "A protocol for user awareness on the World Wide Web." *In Proc. CSCW'96*, Boston, Ma., Nov. 1996, pp. 130-139.

Potts, C., and **Catledge, L.**, "Collaborative conceptual design: A large software project case study." Computer Supported Cooperative Work. Vol. 5, No. 4. 1996, pp. 415-445.

Ramesh, B., V. Dhar. "Supporting Systems Development by Capturing Deliberations during Requirements Engineering", *IEEE Trans. on Soft. Eng.*, Vol. 18, No.6 pp. 498-509, June 1992.

Rodríguez, M. "Organizational Memory for Technical Reviews in Electronic Meeting Rooms." *M.Sc. Thesis* (In Spanish). CICESE. Oct. 1998.

Rumbaugh, J., et al. *Object-Oriented Modeling and Design*, Prentice Hall, 1991.

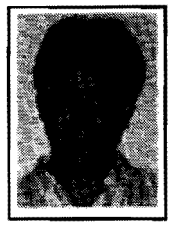
Ruiz, D., Favela, J. "Collaborative review and edition of HTML documents." *4th International Workshop on Groupware*, Rio de Janeiro, Brasil. Sept. 1998, pp. 113-128

Souder, W. "Managing Relations Between R&D and Marketing in New Product Development Projects" *Journal of Product Innovation Management*, Vol. 5, pp. 6-10.

Terveen, L., Selfridge, P., Long M. "Living Design Memory: Framework, Implementation, Lessons Learned." *HCI*, Vol. 10, No. 1. 1995. pp. 1-38

Walz, D., Curtis, B., Elam, J. "Inside a Software Design Team: Knowledge Acquisition, Sharing and Integration." *Comm. ACM*, Vol. 36, No. 10. 1993. pp. 62-77.

Jesus Favela *is a Professor at CICESE in Ensenada, Mexico. He holds a BSc. from UNAM, and a MSc. (1987) and a PhD (1993) in Computer-Aided Engineering from MIT, where he worked as a research assistant at the Intelligent Engineering Systems Laboratory. His research interests include: Computer Supported Cooperative Work, Multimedia Information Retrieval and Software Engineering.*



Josefina Rodríguez *received a B.Sc. degree in Physics from the UABC and a M.Sc. degree in Computer Science from CICESE where she currently works in the Computer Science Department. Her research interests include software engineering and in distributed groups, software processes, and organizational computing.*

Guillermo Licea Sandoval *is a Proffesor at the Universidad Autónoma de Baja California (UABC) in Tijuana. He has a Bachelors degree in Computer Science from the UABC and a Master of Sciences degree from Instituto Tecnológico de Tijuana. He is currently working on his doctoral thesis "A pattern system for the development of synchronous distributed cooperative software" at CICESE. Hi is interested in: cooperative work, object-oriented methods and languages, and software enginnering.*

J. Antonio García-Macías *is a PhD candidate at LSR-IMAG laboratory in Grenoble. Previously, he was an industry consultant in telematics and associate professor in Computer Science. His current research interests include distributed multimedia systems, nomadic networking, adaptable hypermedia, and active networks.*