Distributed Pair Programming on the Web

Hiroshi Natsu, Jesus Favela, Alberto L. Morán, Dominique Decouchant, and Ana M. Martínez-Enríquez

Ciencias de la Computación, CICESE, Mexico
Facultad de Ciencias, UABC, Mexico
LSR-IMAG, Grenoble, France
Departamento de Ing. Electrica, CINVESTAV, Mexico
{hnatsu, favela}@cicese.mx, {Alberto.Moran, Dominique.Decouchant}@imag.fr, ammartin@cinvestav.mx

Abstract

Pair programming is an Extreme Programming practice, where two programmers working side by side on a single computer produce a software artifact. This technique has demonstrated to produce higher quality code in less time it would take an individual programmer. We present the COPPER system, a synchronous source code editor that allows two distributed software engineers to write a program using pair programming. COPPER implements characteristics of groupware systems such as communication mechanisms, collaboration awareness, concurrency control, and a radar view of the documents, among others. It also incorporates a document presence module, which extends the functionality of instant messaging systems to allow users to register documents from a Web server and interact with them in a similar fashion as they do with a colleague. We report results from a preliminary evaluation of COPPER which provide evidence that the system could successfully support distributed pair programming.

1. Introduction

Extreme Programming, also known as XP is a software development methodology that has gained considerable attention in recent years. Its purpose is to simplify and make more efficient the software development process to allow for the development of projects with high risks and constantly changing requirements, where costs and time become crucial. XP involves several techniques and practices that support each other towards these purposes.

One of these techniques, which is being adopted by software development groups is known as Pair Programming, in which two programmers work side by side, on a single computer, to jointly produce an artifact (design, algorithm, code, etc.). It has been reported that this technique can be accounted for the development of higher quality software in half the time it required a single programmer [2,12].

However, pair programming requires the two participants to be at the same location. As the trend towards global software development continues, this limitation becomes more important. In this work we describe the development and test of a synchronous collaborative writing tool, named COPPER (COllaborative Pair Programming EditoR) designed specifically to support pair programming among distributed collaborators.

The rest of the paper is organized as follows. Section 2 describes the pair programming technique and the requirements of a tool to support it in distributed environments. Section 3 presents the design and implementation of COPPER, a synchronous source code editor, designed to satisfy these requirements. In Section 4, we present preliminary results of the use of this tool among a group of computer science students and discuss the major findings of this evaluation. Finally, Section 5 presents our conclusions and future work.

2. Extreme and Pair Programming

Extreme programming, also known as XP (eXtreme Programming), includes a set of principles and practices for the rapid development of high quality software. XP is “extreme” in the sense that it identifies 12 best practices of software development and takes them to an extreme. For instance, if code reviews are a good idea, then code will be reviewed all the time (pair programming); if software testing is good, then it should be done early and often (unit testing). Beck [1] defines XP as “a light methodology for small and medium size
teams developing software with ambiguous or rapidly changing requirements”.

XP was originally developed by Kent Beck with the participation of Ward Cunningham and Ron Jeffries. The methodology was based on the experience they gained developing object-oriented software and their effort to single out the issues that facilitated the development of software.

The practice of XP requires a frequent and effective communication among the members of a development team, which emphasizes the need for team members to be geographically co-located [7]. Our work originates from our hypothesis that the use of appropriate groupware tools could allow distributed teams to effectively apply XP. In particular, we focus on pair programming, a technique used in XP during coding. We look to determine whether this technique can be applied by teams that are physically distributed and yet obtain the same advantages as collocated teams do.

2.1. Pair Programming

Pair programming is not a new term, over the last 20 years the software industry has applied this technique with reported benefits. In this technique, two programmers work together to produce an artifact (design, algorithm, code). The two programmers work as a unit, as a single mind responsible for all aspects of the artifact. One programmer, the driver, controls the pen, mouse, or keyboard to write the code. His colleague actively observes the work produced by the driver, looking for defects, alternatives and considering the implications of the strategy being followed. The pair changes roles periodically. Both participants are active throughout the process and share the responsibility for the work being produced [11].

The successful application of this technique requires the use of an appropriate workplace: “The programmers should be able to sit side by side and program, looking simultaneously at the computer screen, sharing the keyboard and the mouse” [11]. While they work together, the couple should be able to share the keyboard without changing seats.

Extreme programmers require being constantly in touch with their team and for this, their workspaces need to be open and facilitate communication among peers as well as casual and informal encounters.

Several benefits of pair programming have been reported. Among them, an increase in productivity; an increase in product quality due to the higher number of defects found by the continuous review by one of the peers; a reduction in administrative risks derived from the fact that more than one team member is familiar with each module; it increases team coordination and communication; and it improves working conditions since programmers don’t feel isolated [2,12].

2.2 Distributed Pair Programming

One of the main trends in software development has been the globalization of the software industry [7]. Frequently, software developers are required to work in groups that are geographically distributed. There are several motivating factors behind this trend. Among the most important ones [6]:

- Software companies require highly qualified human resources and they look to fulfill this need by hiring programmers in different cities and countries.
- To be closer to the market and have a shorter response time, many companies have placed development groups closer to their client’s location.
- Virtual development groups might need to be created quickly to exploit new market opportunities.
- By working on different times zones development groups can work continuously on critical projects.

Considering this trend towards distributed programming we have aimed at finding out whether distributed programmers could effectively apply the pair programming technique with the use of appropriate groupware tools, and what would be the requirements of such tools. For this purpose we defined a set of requirements, and developed a synchronous collaborative source coding tool named COPPER.

The collaborative authoring of source code can be considered a special case of collaborative writing, a field that has received considerable attention within Computer Supported Cooperative Work (CSCW) community. We have identified the following requirements of a distributed pair programming tool:

R1. Synchronous editing of source code. As is the case for any modern source code editor it should highlight keywords based on the programming language being used and provide conventional editing tools such as: Cut, Copy, Paste, Find, and Replace.
R2. Only two programmers need to collaborate at the same time.
R3. The system should support the options of compiling and executing the source code being edited and should notify the users of the error messages reported by the compiler.
R4. The source code files to be shared should be stored in Web repositories to ensure that documents are available to all members of the development team. Furthermore, configuration control tools are increasingly being developed on top of Web servers to take advantage of the Web’s ubiquity and open standards.
R5. Access to documents being edited should be controlled at the repository. Mechanisms to request and obtain shared resources need to be provided.

R6. Pair programming demands frequent communication between colleagues. The system should support text and audio-based communication. Video is not considered necessary.

R7. Awareness of the presence and state of authors and documents, as well as access rights pertaining to shared resources should be provided to the users.

The next section describes COPPER, a groupware tool developed to address these requirements.

3. The COPPER system

The COOPER system is based on the client-server architecture. It is composed of three subsystems: Collaborative Editor, User and Document Presence (IM&P), and Audio subsystems. On the client side, these three subsystems form an application used to write programs, to access document services and to communicate with peers (see Figure 1).

![Figure 1 Architecture of COPPER](image)

On the server side, COOPER includes an Instant Messaging and Presence (IM&P) server and a Document server. The IM&P server, as it names implies, distributes messages among users and maintains user and document presence. We are using the open source Jabber [9] IM&P server to provide this functionality.

The Document server provides access control to document storage and editing. It also provides information to the IM&P service with regard to the presence and status of the documents it contains. To achieve this, we are using a WebDAV server [4] customized with some extensions to meet our requirements. Further detail on each of these subsystems is provided in the following subsections.

3.1 Collaborative editor

The editor can be used either individually or in synchronous collaborative mode. Users can be either co-located or distributed on the Internet. This subsystem is further decomposed into the Editor module and the Document server.

The Editor module implements a turn-taking synchronous editor (see Figure 2). Document editing is performed in the current Editing window (Figure 3F). The user holding the floor (or editing right) uses this window to work on the currently loaded document.

Common document management and editing functionality is provided by means of the application’s Menu bar and the toolbar (Figure 2A). These operations include: Connect, Disconnect, New, Open, Save, Save As, Cut, Copy, Paste, Search, Repeat Search, and Search & Replace. Help is also provided by these means. Actions performed in the editor are propagated to the collaborator’s client.

As an example of their use, consider the Open button (third button left to right) of the toolbar. When pressed, an Open dialog (Figure 2G) appears, allowing interactions with documents stored in the Document server. This dialog presents an integrated file hierarchy with documents from the local machine and from other distributed WebDAV servers [10]. This allows for seamless navigation and document retrieval from the individual or collaborative work environment. Several documents can be edited at the same time, even if these documents come from different WebDAV servers.

The Document server offers a centralized information repository, which provides:
- Document storage: Stores versions of documents.
- Document editing access control: Manages locks on documents to avoid the lost-update problem.
- User authentication and permissions: Manages user authentication and access control lists on documents to avoid unauthorized accesses.
- Document presence extensions: Manages an IM&P client, which “listens” and informs to the IM&P system the results of operations performed on the documents [8]. We further discussion this topic in the next subsection.

Each editor client “owns” the documents opened by its user, and it is the only one allowed to perform operations, such as Save, Save As, and Close, on these documents. When the connection is broken, each client keeps their “own” documents, and the user can continue working on them in individual mode. While in this mode, clients are ready to send or receive invitations to begin collaborating.

Floor management (or editing access control) is represented using a “traffic light” metaphor (Figure 2B), which is activated when working in collaborative mode.
This component includes an action button to request and grant floor control. The control state of the document is depicted by the colors shown by the traffic light. There are four possible states:

- Red indicates that the user is working as an observer (e.g. reviewing code), and cannot edit the code at the moment.
- Green indicates that the user controls the editing window, and can edit the code.
- Red and yellow indicates that the user is an observer, but has requested the editing control.
- Green and yellow indicates that user has the control on the editing window, but that the other user (observer) has requested the floor.

Available floor control actions depend on the control state. In each state, only one action can be performed. The label of the button shows possible actions. These actions are:

- Request: Demand control of the floor to edit the source code.
- Grant: Release control of the editing window to the other collaborator.
- Cancel: Abandon a request to obtain the floor.

Collaboration awareness is provided by means of a radar view (Figures 2C and 2D), editing window titles (Figure 2E), the status bar (Figure 2H), and the floor or editing control access component (described earlier).

The radar view provides a general overview of the document being edited; it shows document changes in real time. The local user’s viewable screen is shown in one color (Figure 2D). If the collaborator is present, it shows his/her document viewable screen in a different color (Figure 2C). When these two regions overlap, this is marked using a lighter shade. Finally, the radar view also serves as a document navigation tool; selecting a particular line of the document in the radar view causes the current editing window to display the segment of the document where the selected line is present.

Editing window titles (Figure 2E) display the name and location of the document, as well as the identities of the owner of the document the collaborator (if present).

The status bar (Figure 2H) provides information on the last operation performed, as well as on the date and time it was performed. When the system is used in collaborative mode, the information displayed here includes those actions performed by the collaborator.

Communication between the editing module and the document server happens through the HTTP protocol, and its WebDAV extensions. When two editor modules
interact a communication channel (socket) is established to exchange objects describing the events that occur.

### 3.2 User and document presence

The User and Document Presence subsystem extends the functionality of traditional IM&P systems to provide first class presence to documents stored in the Document server. Figure 3 shows the architecture of the current implementation of the User and Document Presence subsystem.

![Figure 3. Architecture of the user and document presence subsystem](image)

The client-side module of the User and IM&P subsystem (Jabber-DAV Client in Figure 4) is used to send and receive messages to/from collaborators, manage the user’s own presence and provide this presence information to other collaborators. Additionally, it extends this functionality to interact with the documents in a similar way as one would interact with users [8]: adding or deleting documents from the document (presence) list, and sending “group” messages to subscribed users of a document. Furthermore, subscribed users receive messages from the IM&P service whenever the document’s availability and status information changes. Finally, this module implements the functionality offered by WebDAV, to perform basic editing operations on subscribed documents (e.g. locking a document to avoid users from concurrently modifying the same document).

The functionality of the IM&P module is accessible by means of several components, as shown in Figure 4. The “traditional contact list” is decomposed into an author list (Figure 4B) and document list (Figure 4E). These lists show availability and status information of subscribed users and documents. Availability options for users include “Online” and “Offline”, while for documents they include “Available”, “Locked” or “In-use”, and “Not available”. These options are shown by means of availability icons (Figures 4C, 4D, and 4F). Additional status information is shown by means of labels associated to the list elements (Figures 4H and 4I). Status options included are “Online”, “Away”, “Extended away”, “Free for chat” and “Do not disturb”, which can be specified by means of the Presence menu (second option in 4G). Own identity and status information are shown in the application’s title bar (Figure 4A) and status bar (Figure 4J), respectively.

![Figure 4. Client of the IM&P module](image)

Document operations are available in the WebDAV menu (fourth option in 4G). Using these operations a user can lock on a document, get it to work on it locally (e.g. using the Editor module), save it back to the Document server and finally unlock it to allow other users to work on the new version of the document.

Lastly, the module can send messages to users directly or through documents. Sending messages through a document corresponds to a “group” message to all users subscribed to that document. Two types of messages can be sent or received, chat and instant messages. Chat messages allow “concatenating” several messages on a single window to present a “conversation”, while instant messages are sent one at a time. As users can establish several chat conversations or receive several instant messages from different users, the name of the user with whom the conversation is held appears in the window’s title bar (Figure 4K). An example of a chat message window containing an ongoing conversation is shown in Figure 4M.

The Instant Messaging and Presence (IM&P) server provides the functionality available on the client-side module: presence management and message exchange, contact lists creation and distribution, as well as personal information configuration. It is also used to
authenticate all user requests to start a session in the system.
The Jabber protocol [9] is used to handle communication between the User and Document IM&P client and the IM&P server. In addition, this module uses the WebDAV protocol to communicate with the Document server.

3.3 Audio subsystem

The Audio module establishes and maintains an audio communication channel between two clients so that their users can hold a conversation while collaborating. In our current prototype, we decided not to implement this module, but to use a public domain tool for these purposes.

4. Testing COPPER

In order to test the COPPER system in a realistic environment and evaluate its effectiveness in supporting distributed pair programming we tested the system with computer science students from a local University. The main objective of our evaluation was to find out if distributed pair programmers working with COPPER achieved a similar productivity as those that were co-located. Before we conducted this experiment, we conducted a preliminary evaluation to test the system’s functionality. As a result of these tests a few modifications were made to the system to ensure its proper functionality.

4.1 Description of the experiment

For the experiment a group of students were asked to solve a computer programming task using pair programming and the COPPER system as development environment. The students were asked to fill a demographic questionnaire before the experiment was conducted and a second one at the end of the programming exercise. In the second questionnaire students were asked to evaluate:

- Collaboration with the colleague. To what extent were the contributions of the collaborator useful to complete the task at hand, how the two coordinated and the difficulties they perceived.
- Usefulness of the tools provided by the editor to modify or create the program.
- The use of the IM&P tool to support communication.
- The appropriateness of the semaphore as a mechanism to provide awareness and floor control.
- The use of the radar view in support of collaboration awareness.

Additionally, several open questions were included for the subjects to describe problems they encountered and propose modifications to the system.

A total of 18 students participated in the exercise, which was part of a weekly programming session scheduled as part of a regular course. Before the experiment they participated in a 20 minute session in which they were introduced to the pair programming technique and were shown the functionality of the COPPER system.

The participants were randomly grouped in 9 couples. Five couples were distributed, one member working in one computer lab and the other in a second one. The other 4 pairs worked side-by-side in the same lab with each pair sharing a single computer. All nine groups used the COPPER system to do their work. The experiment was conducted in a session that lasted 90 min. Students in the distributed groups were asked to use the text-based communication tools provided by the system. Since there were several pairs working in the same room we didn’t install the audio communication module to avoid distractions.

4.2 Results

We present the results of the experiment in three categories: user profile; evaluation of the task performed by the students; and evaluation of the COPPER system

4.2.1 Population. The students that participated in the study were between 20 and 23 years old; 22% of them were female. They were on their third year of college and had considerable programming experience. They had no formal training in collaborative systems, but were avid users of electronic communication tools, notably email and instant messaging. None of the students had previous formal experience with pair programming although a few had heard the term before and 17% of them recognized having a previous experience following a similar process.

We asked them about their perception regarding pair programming. Since they had not experienced it before their answers reflect their perception based on the explanation they were given about the technique. They all agreed that code of higher quality could be generated using pair programming. When asked about the use of human resources and computer equipment 89% thought that pair programming allows a better use of these resources. Finally, 75% of them thought that the use of this technique allows programmers to learn more and
finish their task in less time. Overall the subjects had a very positive opinion of the technique before using it.

4.2.2 Evaluation of the task. In this section we present the results of the second questionnaire that were related to the subject’s perception of the task performed.

When asked about the experience of working with another programmer during the exercise, only 1 person described the experience as being negative, and two reported the experience as uncomfortable. In contrast, the remainder 84% felt comfortable sharing the activity with another student.

The users were asked to evaluate the help they received from their collaborators when they were working as drivers. The results are presented in Figure 5, where it can be seen that the perception of those working in the same site was very positive, while that of those working in distributed mode was less so, but still positive.

![Figure 5. Perceived help received from colleague](image)

Users were also asked to evaluate the adequacy of the communication they had with their peers. The results are presented in Figure 6, showing that the perception was somewhat positive and slightly less so for those who were distributed.

![Figure 6. Communication with the colleague](image)

Even though there was a high degree of participation among the pairs, 44% of the couples reported coordination problems. In this regard they expressed the following opinions:

- “Text communication is slow and one loses a lot of time explaining why the code should be one way or another”
- “The chat window hides behind the editor when you are programming, it should be always on top to know if someone is trying to communicate with you.”
- “We had never worked this way before, it got some getting use to it, but afterwards everything was fine”
- “We didn’t agree on how to write the program”

4.2.3 Evaluation of the COPPER system. Users were asked to evaluate the system based on its ease of use as a source code editor and the mechanisms it provides to facilitate collaboration. Only 6% of the users found COPPER to be of little use; half the users evaluated the tool as an average editor, and 33% found it to be very useful. When asked to compare the editor in COPPER with the platform they commonly use to program, 6 users thought it was better, 9 said it was about the same, and two of them considered it to be worst.

With respect to the use of instant messaging and chat to support communication among the distributed couples, 4 users reported it as being very good (40%), one said it was good, two more found it to be average, and two more below average. Those who had problems communicating with this tool found the text-based communication to be inadequate to support an activity as dynamic as pair programming. Another main concern was that the chat tool appeared in a separate window from the editor, and often the chat window will be hidden by the editor not allowing the user to be aware of the arrival of new messages. Even with these restrictions however, the users were able to interact through the mechanisms provided.

The five couples that were distributed also evaluated the collaboration facilities provided by the system. Figure 7 shows the perceived value of the radar view as an awareness mechanism and the semaphore for floor control. The results show that both tools where considered useful.
The users were also asked to provide additional comments about the system. These included:
- “It was entertaining; it was actually very practical to work at a distance.”
- “It was fun to watch my colleague write the program, detect his errors and request control of the system to correct them”
- “It is a very nice system since one can work with someone else to program more rapidly”

4.3 Discussion

From the results of the experiment we observe that the users experienced few problems working with COPPER since the interface and mode of operation of the system is similar to the editors they are familiar with.

Several users reported problems of coordination with their colleagues. This was to be expected since they had no previous formal experience with pair programming and some adjustment needs to take place for them to get accustomed to work in pairs (pair jelling) [11].

Even with the limitations of the system most users (83%) considered COPPER to be equivalent or better to the development environments they currently use. They highlighted the system’s groupware features that allowed them to work with a remote colleague. Moreover, the results of the groups that were co-located when using the system to pairs that worked remotely without observing significant differences in performance.

Nevertheless, the users of the system pointed at several limitations of the tool, chief among them, the limitations of a text-based communication tool to support an activity as dynamic as programming. We plan to work on these findings to improve the system’s functionality. Additionally, we expect to change the shared document storage facility to support a more robust platform as proposed in [3].

Acknowledgments

This work was partially funded by ECOS and Anuies under the grant No. M98M01 and by CONACyT under grant U-40799, which also provided scholarships for the first and third authors.

5. Conclusions

Extreme programming techniques, and in particular pair programming, are gaining considerable attention given their advantage at handling software development projects with vague or changing requirements. As the software industry continues to grow and their practice becomes global, distributed teams will require appropriate tools to support their software development practices.

With this motivation we have designed and implemented a synchronous groupware tool to support distributed pair programming. The system uses a WebDAV server as a repository for shared source code, several collaboration awareness tools and a communication tool that extends the traditional instant messaging functionality to provide awareness of, and means for operating with, documents and other shared resources stored in a web server.

Our preliminary evaluation of the system provides sine evidence of its effectiveness for supporting distributed pair programming. We compared groups that were co-located when using the system to pairs that worked remotely without observing significant differences in performance.

6. References


