UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MEDELLÍN
FACULTAD DE MINAS

DYNA
http://dyna.medellin.unal.edu.co/

Assessing the awareness mechanisms of a collaborative programming support system

Ana Isabel Molina [a], Jesús Gallardo [b], Miguel Ángel Redondo [c] & Crescencio Bravo [d]

[a] Escuela Superior de Informática de Ciudad Real, Universidad de Castilla-La Mancha, Spain, AnaIsabel.Molina@uclm.es
[b] Escuela Universitaria Politécnica de Teruel, Universidad de Zaragoza, Zaragoza, Spain, Jesus.Gallardo@unizar.es
[c] Escuela Superior de Informática de Ciudad Real, Universidad de Castilla-La Mancha, Spain, Miguel.Redondo@uclm.es
[d] Escuela Superior de Informática de Ciudad Real, Universidad de Castilla-La Mancha, Spain, Crescencio.Bravo@uclm.es

Received: February 18th, 2015. Received in revised form: March 24th, 2015. Accepted: September 29th, 2015.

Abstract
The learning and teaching of Programming can benefit from the principles of *Computer Supported Collaborative Learning* (CSCL). With that purpose in mind, the COLLECE system was created to support synchronous collaborative programming in learning settings. Unlike other systems with similar objectives, COLLECE incorporates many elements to support group awareness. This article presents an empirical study in which the usefulness of some of the awareness mechanisms included in this system is evaluated. One of the main contributions of this work is the combination of different techniques to evaluate interactive systems, such as questionnaires, laboratory testing, heuristic evaluation, automatic logging and eye tracking techniques. The joint use of these techniques (of both a subjective and objective nature) allows us to carry out a more complete analysis of the system under study and, in particular, about its support of awareness.

Keywords: Collaborative programming; evaluation; usability; groupware, awareness; eye tracking.

Evaluando los mecanismos de awareness de un sistema de soporte a la programación colaborativa

Resumen
El aprendizaje/enseñanza de la programación puede beneficiarse de los principios del *Aprendizaje Colaborativo soportado por Computador* (CSCL). Con el objetivo de soportar tareas de programación colaborativa distribuida síncrona se creó el sistema COLLECE. A diferencia de otros sistemas con objetivos similares, COLLECE incorpora una gran cantidad de elementos de soporte al awareness. En este artículo se describe un estudio empírico en el que se evalua la utilidad de algunos de los mecanismos de awareness incluídos en este sistema. Una de las principales aportaciones de este trabajo es la combinación de varias técnicas de evaluación de sistemas interactivos (cuestionarios, testing en laboratorio, evaluación heurística, logging automático y técnicas de seguimiento ocular). El uso conjunto de todas estas técnicas (algunas objetivas y otras subjetivas) permite realizar un análisis más completo del sistema objeto de estudio y, en particular, de su soporte al awareness.

Palabras clave: Programación colaborativa, evaluación, usabilidad, groupware, awareness, eye tracking.

1. Introduction

The advantages that have arisen lately in the area of telecommunications and the Internet have allowed the principles of *Computer Supported Cooperative Work* (CSCW) [1] to be applied in several fields; programming learning is one such field. In *collaborative programming* two or more members of a team collaborate synchronously on the same programming task from geographically distributed locations, using generic or specific groupware tools.

In order to support this activity, the COLLECE tool (*COLLaborative Edition, Compilation and Execution of programs*) was developed [2]. COLLECE allows for the editing, compilation and execution of programs written in Java and C languages. This system has been evaluated in several settings with different goals. For example, the quality

of the products obtained by a group of programmers has been analyzed, together with the process that results in these programs as final solutions to a problem [3]. We have also carried out some evaluation studies based on users' -who have made use of the system- opinions. We have worked with programming students as well as with developers from a software company [2].

One of the most important features to consider when developing groupware systems is their *awareness* support [4]. Awareness can be defined as the perception of the activity the members of a group are undertaking and their knowledge about it. The use of techniques for awareness support allows for a context to be provided on the activity itself, and this improves the effectiveness and efficiency of the group work [5].

In this work we focus on analyzing and evaluating the *awareness* mechanisms provided by the COLLECE system. Unlike previous evaluations, which were mainly based on the subjective opinion of users, in this work we propose to complement such information with that which is provided by an eye tracker device. The concept of *eye tracking* refers to a set of technologies which monitor and record the way a person looks at a particular scene or image, and, specifically, in what areas he/she focuses his/her attention, for how long and in which order he/she visually explores the material provided. The eye tracking technique has been applied in various disciplines and areas of study: marketing, advertising and evaluation of user interfaces (including web pages) [6,7], etc. By applying this technique, evaluators can obtain objective measurements of a physiological nature about users' visual behavior when they interact with an interactive application. In addition, we can complement the data provided by subjective sources of information (for example, the learner's subjective perception collected by questionnaires about his/her satisfaction) and contrast it with this objective source of information.

The remainder of the paper is structured as follows: in the following section we review the main works related to the field of collaborative programming, as well as some techniques for the evaluation of collaborative systems. Next, in Section 3, we describe the features of the COLLECE system, focusing on its awareness support. In Section 4 we move on to describe the results of the empirical study carried out, as well as the results obtained. Lastly, in Section 5, we present the conclusions derived from the work and the following lines of research.

2. Background

In this section we review some related work and present some foundations that will be useful for an adequate understanding of the work we describe in this article. First, we talk about awareness and its support in collaborative programming systems. Later, in Section 2.3, we talk about techniques for evaluating the usability of collaborative systems as well as their awareness support. Section 2.4 deals with the use of eye tracking techniques toevaluate these aspects more objectively.

2.1. Awareness support in collaborative systems

Collaborative systems introduce new concepts and requirements, such as shared workspaces and collaboration protocols, which do not exist in single-user systems. Perceiving and understanding the responsibilities, activities and intentions of other members of a collaborating ensemble is a basic requirement for group interaction [8]. In face-to-face interaction, it is easy for collaborators to establish a shared background of understanding about who else is present in a workspace, what other collaborators are doing, and so on. However, when group members are geographically spread out, maintaining awareness of group members is much more difficult. In order to facilitate group collaboration effectively, groupware systems must provide group awareness support. *Group awareness* is defined as ''an understanding of the activities of others, which provides a context for your own activity'' [4].

Many techniques have been applied in several systems to support group awareness. The more commonly used mechanisms of capturing and presenting awareness information are the called *2D on-screen awareness mechanisms.* This group includes the WYSIWIS (*What You See Is What I See*) technique, and the incorporation of telepointers, radar views, multi-user scrollbars, or fisheye views [9,10] in graphical user interfaces of groupware systems. All these mechanisms are based on capturing and tracking keyboard events, mouse events or viewports to obtain information about collaborators' activities in a shared workspace. Other systems incorporate *audio and video-mediated awareness mechanisms*. The use of audio and video channels is very useful in supporting communication between members of a group. In recent years a new category of awareness mechanisms has gained the attention of researchers: *sensor-mediated awareness mechanisms*. This approach proposes the use of specialized sensors, visual signals and devices to support group awareness. Examples of such tools include eye tracking, electronic badges and sensors or even wearable appliances [11].

2.2. Systems for supporting Collaborative Programming

Computer Programming is a complex and creative task that can take advantage of collaborative environments and thus be supported and enhanced using groupware systems and distributed architectures. Collaborative programming allows distributed programmers or students that are learning this discipline to work together on the same program or software application. *Collaborative Programming* is a promising tool to scaffold the collaborative learning of Programming, especially since Programming is a difficult subject for students to learn and for teachers to teach. Johnson [12] points out that the process of analyzing and criticizing software artifacts produced by others is a powerful method for learning programming languages, design techniques and application domains.

There are a great number of works that have faced the challenge of supporting *Distributed Collaborative*

Programming. For example, RECIPE (*REal-time Collaborative Interactive Programming Environment*) [13] allows programmers who are geographically spread out to concurrently participate in the design, implementation, testing, debugging and documentation of a program. In order to achieve this, RECIPE allows for the conversion of single user compilers and debuggers in collaborative applications. In the same way, it allows for the integration of existing collaborative editors into the system. However, it does not provide specialized tools for the communication among programmers, and it also lacks some awareness support tools due to its high coupling. Another environment that supports the edition, compilation and execution of programs is DPE [14]. This system includes some communication channels, both textual and audio-based. However, DPE also has limited support for awareness and task coordination. Other similar approaches have attempted to integrate collaborative support in the *Eclipse* environment by making use of plug-ins. One of the most relevant works in this field is *Jazz Sangam* [15], which, among other functionalities, includes instant messaging and version control.

Such systems usually have some limitations regarding awareness support due to their high coupling and to the lack of specific tools for coordination and communication. Thus, the presence of elements for awareness support is one of the most relevant defects of most systems that support collaborative programming, even when it is an essential feature for improving the experience of the collaborative work. Therefore, one of the points on which we focused when developing COLLECE, was the suitable awareness support, as we will discuss in Section 3.

2.3. Techniques for evaluating Collaborative usability and awareness support

One of the main needs that arise once a collaborative system has been implemented is to evaluate its usability and the support it gives to the collaborative activity.

Different methods have been proposed to evaluate *usability*, but they mainly refer to single-user systems. These methods include consistency inspection techniques, techniques for inspecting standards, the use of cognitive walkthroughs and heuristic evaluation [16]. However, these techniques, although they are useful up to a point, are not the most appropriate for the evaluation of groupware systems. Therefore, other techniques are used to evaluate the usability of this type of systems. The work carried out by Pinelle and Gutwin [17] suggests that *groupware usability* be defined as the "extent to which a groupware system allows teamwork to occur –effectively, efficiently and satisfactorily– for a particular group and a particular group activity". Some of the new techniques for groupware usability evaluation are basic inspection methods, cognitive walkthroughs adapted to collaborative systems [17] and an adaptation of the Nielsen heuristics for application to groupware systems [18].

In this work we are interested in the *evaluation of awareness support* of collaborative systems. There are several works that address this issue. In Convertino et al. [19], the authors carried out some evaluation studies about *activity awareness*. This has been achieved by means of some tasks in which whether the mechanisms perceive both synchronous and asynchronous work is tested. The suitable choice of these tasks, which should copy the real work carried out with the tools, is the key to performing a valuable evaluation. In general, there are several problems that must be faced when evaluating awareness support.On the one hand, the first problem is the fact that the reception of awareness information may generate an interruption in the work of the user, and on the other hand the second is the possible interference in the privacy of the user that may occur when managing and visualizing such information. Both problems are analyzed in works such as the one by Röcker and Magerkurth [20], in which they try to find a different approach to the design of user interfaces that solve those problems. Lastly, we want to mention the checklists proposed by Antunes et al. [21], which can be very useful for software developers when examining the quality of the awareness support in their developments.

Several authors have tried to formalize the different concepts relating to awareness by proposing theories, frameworks and taxonomies and have tried to help developers and evaluators when considering these aspects in the development and evaluation of collaborative systems. One of the most outstanding contributions in this field is the *Theory of Awareness* by Gutwin & Greenberg [22], which includes a framework that defines different awareness elements and makes the validation of awareness support possible by means of a set of relevant questions. The main contribution of the work by Gutwin and Greenberg has been to identify the elements of knowledge that make up the core of *workspace awareness*, each one relateing to the question answered in order to provide that element of knowledge.

However, most of these awareness evaluation techniques are based on heuristic frameworks and the use of questionnaires and checklists completed by experts (in the case of heuristic evaluation) or by system users. Most of these techniques are highly subjective and, therefore, the results obtained in their application are highly subject to biases. In this paper we propose complementing such evaluation with more objective techniques, in particular, with eye tracking techniques [6].

2.4. Using Eye tracking techniques to evaluate collaborative systems

Interest in the use of eye tracking techniques has increased in recent years as a means of understanding and analyzing the visual attention of users. Eye tracking sessions allow us to draw conclusions about the behavior of visual exploration that users perform when interacting with a software system. Since those measures are of a physiological nature, the results are less subject to biases and cannot be controlled by users. This analysis technique has been used successfully in usability studies of single-user interactive systems, mainly web pages [6,7].

When we carry out an eye tracking sessions, we can perform a quantitative analysis of the test results or a qualitative one. In this last case, we can use two representations: *heat maps* and *scan path* or *gaze plot*, which allows us to graphically show the visual behavior of a user or a group of users. If we want to perform a quantitative analysis, we can use several ***metrics*** that we can collect by means of the eye tracker device [7]. These metrics can be used to determine the areas of the interface where users focus their attention or the cognitive effort involved when they try to understand the visual information provided. Before obtaining the metrics it is necessary to define the so-called *areas of interest* (AOI) of the image displayed. The definition of these AOIs depends on the specific task that is to be performed, and it delimitates those areas of the displayed image or user interface for which we want to obtain the metrics.

Most metrics collected are related to the number and duration of the so-called *fixations* (when the eye remained stationary, focused on an AOI for a certain period of time). It is necessary to point out that a metric can be interpreted in different ways. For example, longer fixations can mean that a user found a particular area interesting but it can also mean that they found the area difficult to interpret. Hence, it is important to attempt to supplement eye tracking data with additional information gained from the participants about their experiences during the activity.

Recently, eye tracking research has witnessed a leap from the single-user scenarios to multiple user, collaborative domains. New concepts have appeared, such as the so-called *dual eye tracking* [23] or remote transfer of points on which the user is focusing his/her attention (called *gaze transfer* [24]).

We can see, therefore, that there is great potential in using this new source of information for evaluating interactive systems in general, as well as collaborative systems, more specifically. Using the metrics provided by the eye tracker allows for contrast and complements other information sources commonly used for assessing these systems. In our case we will use it to determine the usefulness of the different awareness mechanisms included in the COLLECE system, whose main features are detailed in the following section.

3. A collaborative programming CSCL system: COLLECE

The COLLECE (*COLLaborative Edition, Compilation and Execution of programs*) system [2] allows users to edit a program or code fragment, to compile it and to run it collaboratively. Up to now, the languages supported have been Java and C. Because the system is primarily used for teaching-learning purposes, two different actors are recognized: teacher and student. The teacher defines the work sessions and arranges the users participating in them by using management tools. A session is defined by name, type, file containing the formulation of the problem to be solved and schedule in which the session has to be carried out.

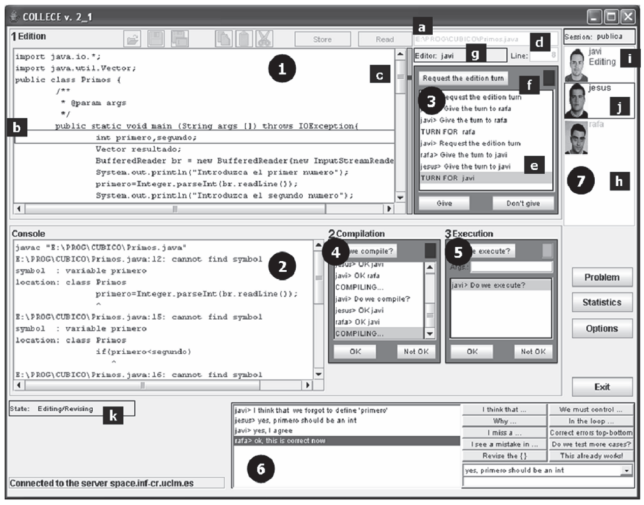

Figure 1. The COLLECE user interface.
Source: [2].

In order to carry out the programming tasks, an explicit collaboration protocol must be followed. First, the students create a program using the collaborative editor (Fig. 1, ❶). They are then able to compile the program, receiving a list of compilation errors (Fig. 1, ❷). Finally, they can execute the program providing that a compiled program is available. Iterations are possible between these three tasks. However, despite this script, the students are free to make their own decisions on when to edit, compile and execute, as well as to decide who is responsible for each task. To do so, coordination tools are available in the workspace to regulate the navigation through the collaboration protocol.

Such coordination processes are modeled with a simple protocol of actions extracted from language. In order to regulate the edition turn assignment (Fig. 1, ❸), we identified the following acts: *Request* the edition turn, *Give* and *Don't give*. With these acts, a user can request the edition turn, and his/her fellow users can express his/her agreement or disagreement. When all the users in the group agree, the assignment is made. Similar actions are used for coordinating when to compile and when to execute the program (Fig. 1, ❹ and ❺, respectively). These coordination tools support multiple proposals, that is to say, proposals coming from more than one user. As a result, listscontaining the historical proposals are necessary as they enable a user to select the proposal to which he/she wants to respond.

The communication during the tasks takes place by means of a structured chat (Fig. 1, ❻). This chat is structured because it offers a pre-established set of communication acts, aimed at providing explicit communication acts that encourages the users' participation, reducing the writing load and focusing the users on the task. Apart from the so-called structured messages, the chat also provides the users with free text messages and the possibility of selecting one of the last messages sent in order to reuse it.

In the final COLLECE user interface four main areas are identified: the edition area at the top (Fig. 1, ❶ and ❸), the console in the middle (Fig. 1, ❷, ❹ and ❺), the chat at the

bottom (Fig. 1, ❻) and the session panel on the right (Fig. 1, ❼). Two functions of the system allow users to consult the formulation of the problem as well as the compilation statistics. The former shows a textual description of the problem to be solved. The latter displays an ordered list of compilation errors and their frequency, so that the students are aware of their more frequently made mistakes.

Besides coordination and communication tools, awareness support is also available in COLLECE. COLLECE deals with the problems of awareness by providing a number of mechanisms to provide information about people, their state and their actions. Specifically, awareness in COLLECE is made possible by means of a number of elements: (i) session panel (Fig. 1, h); (ii) global state (editing, compiling or executing) (Fig. 1, k) and individual state (Fig. 1, i); (iii) tele-pointers, in the form of a colored rectangle drawn around the source code line (Fig. 1, b); (iv) lists of interactions (Fig. 1, e); (v) semaphores (Fig. 1, f); (vi) beeps, when actions occur; (vii) users' position in the source code (Fig. 1, c); and (viii) other mechanisms (Fig. 1, a, d, g and j) such as rectangles and labels to highlight the user and show some elements of information.

The system, developed using Java technology, operates on client/server architecture. The data management as well as the synchronization services for implementing the synchronous collaboration is centralized on a server, whereas the distributed clients (the users executing the system) access the system from a web page.

As was stated in the introduction, COLLECE has been evaluated several times and from different points of view. However, most times the source of information used has been one of a subjective nature. That is, the evaluation has been based on the opinion of students or software developers, questionnaires being the most commonly used technique. Therefore, in this work we propose to complement those evaluative works with additional information of a more objective nature that may be collected during the collaborative edition activity. Thus, in this evaluation we will combine techniques such as questionnaires about subjective perception, heuristic evaluation, testing and automatic logging techniques, retrospective thinking aloud (RTA) and, lastly, eye tracking techniques applied in the context of a usability laboratory.

4. An empirical evaluation of COLLECE

In this section we are going to describe the details of the experience carried out to evaluate the different elements of awareness support included in COLLECE. We have focused on evaluating the *usefulness* for the users of each one during a session of collaborative programming. In order to achieve this, we have used several sources of information. Next, we are going to describe the sample that took part in this experience, as well as the task they had to perform. We will then talk about the design of the experience and the results obtained in it.

4.1. Participants

Ten subjects took part in the experience. All of them were students on the *Systems for Collaboration* course: a subject in the fifth year of Computer Science studies of at the University of Castilla-La Mancha (UCLM). All of them voluntarily agreed to participate in the experimental task.

4.2. Experimental task

The task to be carried out by the participants consisted in modifying a Java program provided to them at the beginning of the session. The program made use of the ICE (*Internet Communication Engine*) framework in order to create some client and server threads, so that the task had an intermediate complexity. Participants had a maximum time of 15 minutes to finish the activity. Five groups of two students were created randomly.

4.3. Laboratory settings and equipment

The experience was carried out at the Usability Lab, owned by the CHICO research group at the UCLM. The laboratory includes, besides the common resources for any computer lab, the proper equipment for usability and accessibility testing of interactive systems:eye tracking testing equipment, several testing and interview rooms (equipped with cameras, microphones and a PA system) and an observation room for monitoring tests. The equipment used for eye tracking was a Tobii X60 model[1]. In order to design, carry out and subsequently analyze the eye tracking session, Tobii Studio 3.0.2 software was used.

Before performing the final evaluation, a *pilot test* was carried out. Four professors and researchers participated in it. As a result of the pilot test, some eye tracking testing conditions which can interfere with the session were controlled. Thus, the testing room was prepared to carry out the final test. This phase allowed us to decide which eye tracking measurements to consider in the final test and how to interpret of each of them.

During the experimental task, two computers were used, one for each member of the pair. Both computers were connected to the Internet. Even when both users were in the same room, they could not see the screen of the other member. Moreover, they were not allowed to talk to each other. Instead, they had to use the communication and coordination tools that are integrated in COLLECE. As suchThe scenario of distributed collaborative programming to which COLLECE gives support was, therefore, more accurately recreated. One member of the group used the computer that had been equipped with the *eye tracking* device, whereas the activity of the second member was recorded by means of a camera pointing at his/her screen (Fig. 2). Both recordings (the one by the eye tracker and the video recording) were synchronized.

[1] http://www.tobii.com

4.4. Experimental procedure

During the development of this empirical study three stages were followed: filling the pretest, intervention or test phase (realization of the experimental task) and posttest. Each pair of participants was summoned at different times to perform the task. The maximum duration of the test for each group was 15 minutes. The methodological recommendations of Nielsen and Pernice [25] were followed for the design and further development of eye tracking tests.

4.5. Data collection techniques

Data from the experiments were gathered using the following techniques:

- **Screen and video recording**. The subjects' monitor screens were recorded. Also, all interactions with the application involving the subjects in the shared workspace were captured. COLLECE includes a module for the analysis and recording of the collaborative editing process. This module allows us to make an automatic *log* of the main interactions performed by workgroups and calculating frequencies of use of the main functionalities of the system. Examples of such interactions included programming events (e.g. editing, compilations, executions), communicative events (e.g. text messaging), and coordination activities (e.g. turn changes).
- Questionnaires. The subjects were asked to fill out several questionnaires. Before proceeding with the experimental task, participants filled out the pretest in paper form. This questionnaire aimed to determine their user profile, for which several items that should be scored on a Likert scale (1-5) were included. Subjects indicated their level of agreement or disagreement with eight statements, which were designed to measure their level of programming knowledge and level of expertise in Java and ICE; their level of theoretical knowledge on collaborative systems and the techniques for supporting awareness; and finally, his/her experience in the use of collaborative tools, collaborative programming tools and the use of the COLLECE tool.

Then, students scored the level of *perceived usefulness* (**PU**) and *intention to use* (**ITU**) of each of the awareness mechanisms included in the COLLECE system (as described in Section 3) on a scale of 1 to 5. These last aspects can be measured by applying the **TAM evaluation framework** (*Technology Acceptance Model*) proposed by Davis [26]. TAM is one of the most widely accepted theories among information system researchers to study the system-acceptance behavior of users [27]. TAM was the first model to mention psychological factors affecting computer acceptance. In this work we have adapted some of the questions proposed by this framework in order to determine the subjective perception of students about the use of the different awareness elements supported by COLLECE.

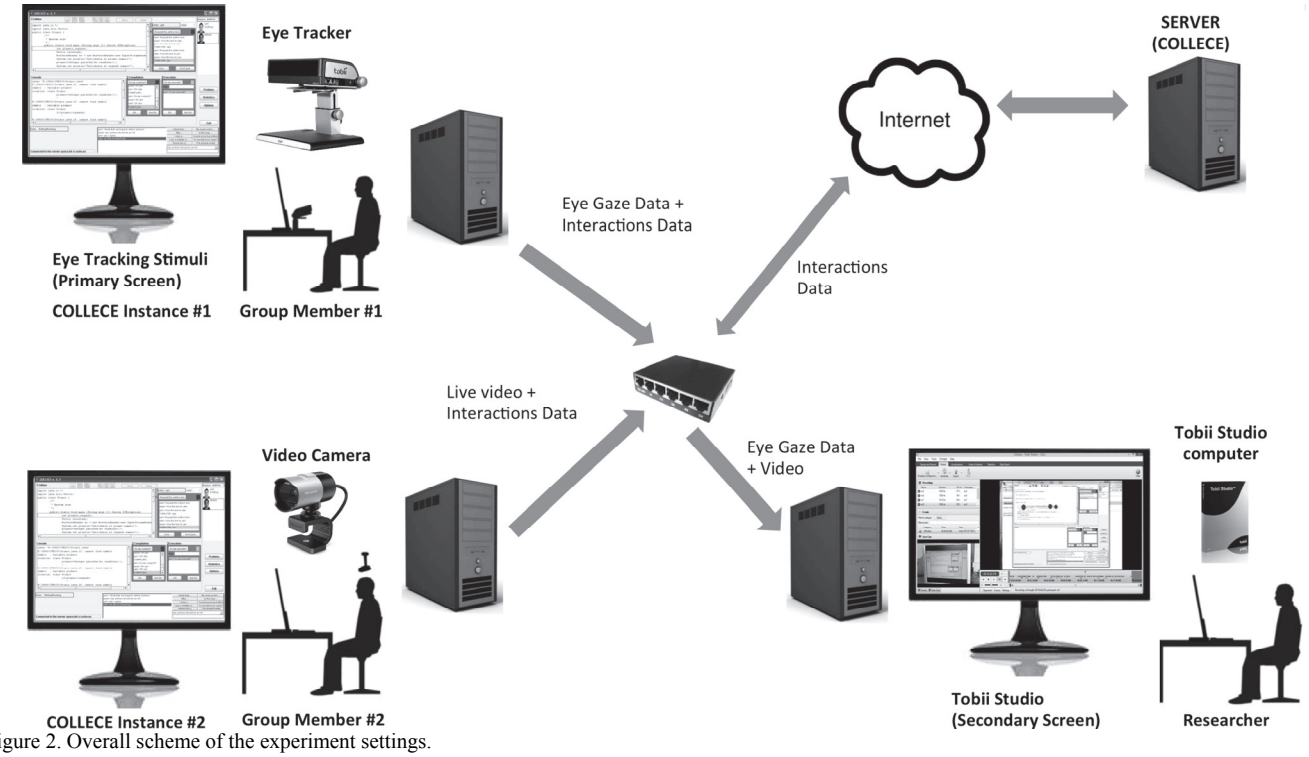

Figure 2. Overall scheme of the experiment settings.
Source: The authors.

Once the pairs of students finished the modification and extension of the program task, they went on to individually complete the **posttest**. In this last phase, students had to score, using the same scale that was supplied before the test (adapted from the TAM evaluation model), the *perceived usefulness* (**PU**) of the main awareness elements supported by COLLECE, but considering, in this case, their use during the realization of the experimental task.

Finally, and taking into account the profile of the participants (students from the *Systems for Collaboration* course), the ten participants adopted the role of experts in groupware user interfaces and moved on to perform a **heuristic evaluation** of the COLLECE system, making use of the dimensions and sub-dimensions that the framework of Gutwin and Greenberg includes [22]. This last questionnaire included twelve questions. Each question offered the experimental subjects a choice of five alternative answers ranging from 1, "very poor support" to 5, "very good support". The purpose of the five-point scale questionnaire was to determine the support provided by COLLECE in terms of workspace awareness. In Table 1, the questions included in this last questionnaire are shown.

- **Eye tracking**. After the pretest was completed, the pairs of students moved on to the test or **intervention phase**. Before beginning, the eye tracking device was calibrated for one of the members of the pair. It must be noted that not all people can participate in a task of this type. This is because there are problems with calibration when the participant uses bifocal glasses, contact lenses or if the lighting conditions are not appropriate. These factors affect the reliability and validity of the data obtained, and, as such, subjects who use vision aids should be eliminated from the final sample. The Tobii Studio software allows us to identify problems during the tracking of the subjects by showing a percentage that indicates the precision and quality of the samples obtained. In the study conducted almost all participants had an accuracy level of about 90%, so it was not necessary to remove any of them from the final sample.

Table 1.
Dimensions, sub-dimensions and questions proposed by the Gutwin and Greenberg framework.*

Dimension	Sub-dimension	Questions
Who	Presence	Is anyone in the workspace?
	Identity	Who is participating? Who is that?
	Authorship	Who is doing that?
What	Action	What are they doing? What are their current activities and tasks?
	Intention	What goal is that action part of?
	Artifact	What object are they working on?
	Changes	What changes are they making? Where are changes being made?
	Activity Level	Are they active in the workspace? How fast are they working?
Where	Location	Where are they working?
	Gaze	Where are they looking?
	View	What can they see?
	Reach	What can they reach?

Source: Adapted from [22].

In an eye tracking session we can extract a large number of metrics that can be interpreted such as measures of interest, cognitive load, emotional arousal, etc. As we have mentioned above, we are interested in measuring the *usefulness* of the different mechanisms for supporting awareness included in the COLLECE user interface. The two metrics provided by the eye tracker device to measure user interest or attention in a certain area of the image (AOI) are the *number of fixations* (**#Fij**) and the *inspection time* of an AOI. Since in the activity solved by the students there was no time limit, it is more appropriate to consider, instead of absolute times, relative times, i.e. the *percentage of inspection time* (**%Insp**) spent by each subject to inspect each of the representations with respect to the total time devoted to analyzing the entire image.

- **Observation.** The experiments were closely monitored by an observer placed in the control room with the support of observation monitors.
- **Retrospective Thinking Aloud (RTA).** As we have mentioned above, an eye tracking metric can be interpreted in different ways. Hence, it is important to attempt to supplement eye tracking data with additional information gained from the participants about their experiences during the activity. To decide upon the final interpretation of each metric, it is useful to apply a retrospective thinking aloud (RTA) session, which is commonly used for usability testing. In this method, once the tasks were completed by participants, they replayed the eye tracking session again and verbalized their experiences while doing the tasks. The application of such method helped us to determine the most suitable interpretation for the metrics gathered during the eye tracking session.
- **Interview.** Also, we used some open-ended questions in interviews, the purpose of which were to ascertain the opinion of participants about the strengths and weaknesses of the COLLECE system, focusing mainly on its user interface and its awareness support. Also, students could make proposals to improve the user interface of the system.

4.6. Results and discussion

In this section we analyze and interpret all the information gathered throughout the task. As noted, we obtain values from different sources. Some of them are of an objective nature (total time taken to perform the task and the metrics provided by the eye tracker), while others are obtained from the questionnaires provided (level of knowledge, perceived usefulness, intention to use, etc.).

In Table 2 we show the values relative to the **profile** of the users who participated in the task. The mean values of the responses given by the participants are shown. As we can see in the table, participants had little experience in the use of the tool (M = 1.70), but had an average understanding of the theoretical foundations of CSCW (M = 2.80) and *awareness* (M = 2.60). Most of them believed that they were good at programming (M = 3.90) and the use of the Java language (M = 4.00), but not a very high level of knowledge in ICE (M = 1.60).

Table 2.
Profile of participants in the empirical study.

	Score*
Level of knowledge in programming	3.90 (0.57)
Level of knowledge in Java	**4.00 (0.82)**
Level of knowledge in ICE	1.60 (0.52)
Theoretical level of knowledge about collaborative systems	2.80 (0.63)
Theoretical level of knowledge about awareness techniques	**2.60 (0.84)**
Level of practical experience in the use of collaborative tools	2.80 (0.63)
Level of practical experience in the use of collaborative programming tools	1.90 (0.88)
Level of practical experience in the use of COLLECE	**1.70 (0.82)**

* We show the mean and, in parentheses, the standard deviation.
Source: The authors.

Although, as discussed in Section 3, COLLECE includes a great deal of awareness elements, we will focus our analysis only on some of them. We defined six areas of interest (AOI) in the user interface of COLLECE, corresponding with six of the awareness mechanisms enumerated in Table 3.

Table 4 shows the ratings that the ten participants gave the main awareness techniques, before (pretest) and after (posttest) of the experimental task, according to the dimensions of the **TAM evaluation framework** [26]. The two dimensions that we want to contrast are those that measure the *perceived usefulness* (PU) and *intention to use* (ITU). Considering the answers given by users in the pretest, we can see that what most of them consider as most useful (PU).What they think they will consult more (ITU) is the session panel, the number of the line of code being edited by another group member and semaphores, which allow them to coordinate and make decisions. Comparing the responses given on the pretest to the posttest we can see that, once the task was completed, users still considered the session panel and semaphores as the most useful elements (PU) (M = 4.50). However, they believed the line number on which another group member is working was not useful during the activity (M = 3.30). As for the multi-user scrollbar, its evaluation in terms of usefulness improves once participants have passed the test phase.

Table 3.
Definition of the areas of interest (AOI) associated with the main elements of awareness of COLLECE.

Area of Interest	Awareness support mechanisms included in COLLECE
AOI-Semaphores	Visual indicator in the coordination tools so that the users can easily perceive (with a green light) when and where there are other users' proposals still awaiting an answer.
AOI-Multi-user scrollbar	Multi-user scrollbar shows the local user's position and remote users' positions in the source code at the same time. The viewport of each user is represented in multi-user scrollbars as a colored bar locating on the right-hand side of the window.
AOI-Session Panel	Visualization of the users who are participating in the programming activity, identified by means of a specific color.
AOI-Editor	Visual information about who is editing.
AOI-Global state	Visual indicator pertaining to the current state of the activity (editing, compiling or executing).
AOI-Code line	Visual indicator pertaining to the number of the code line being edited.

Source: The authors.

Table 4.
Ratings of the *awareness* elements of COLLECE (TAM *framework*)**

Area of Interest (AOI)	Perceived Usefulness (PU) - Pretest*	Intention To Use (ITU) - Pretest*	Perceived Usefulness (PU) - Posttest*
AOI-Session Panel	**4.50 (0.71)**	**4.50 (0.71)**	**4.50 (0.85)**
AOI-Editor	4.20 (1.03)	4.30 (1.06)	4.30 (0.82)
AOI-Multi-user scrollbar	4.00 (1.33)	3.70 (1.34)	4.40 (1.35)
AOI-Semaphores	**4.50 (0.71)**	4.30 (0.82)	**4.50 (0.71)**
AOI-Global state	3.90 (0.88)	3.80 (1.03)	3.50 (0.97)
AOI-Code Line	**4.50 (0.71)**	4.40 (0.70)	3.30 (1.64)

* We show the mean and, in parentheses, the standard deviation.
** Values for n=10 participants.
Source: The authors.

Table 5.
Some use and interaction counts (automatic logging module of COLLECE).

Session ID	#TC	#Comp	#Exec	#MessInter	#Contributions different to "*I think…*"
sc_exp_a	4	2	1	29	5
sc_exp_b	1	1	0	12	2
sc_exp_c	2	2	0	9	1
sc_exp_d	4	5	3	28	4
sc_exp_e	3	2	0	34	7

Source: The authors.

As it is stated in Section 4.5, COLLECE includes a module for recording all the interactions produced during the collaborative editing process. The utility of this information is greater the larger the sample and number of tasks performed by users are; however, this information is not particularly significant in this first evaluation, given the small number of groups participating in the activity. Still, this module allowed us to record the number of exchanged messages (#MessInter) by the five working groups, the number of compilations (#Comp), executions (#Exec) and turn changes (#TC) made by users (Table 5). This module also allowed us to verify that the groups did not make proper use of the *sentence openers* included in the *structured chat*. They used, in most cases, the opener "*I think …*" to start their contributions in chat. This behavior, identified in other previous studies, calls into question the utility of incorporating the *structured chat* in COLLECE and, particularly, in the task under consideration.

We will now move on to discuss the results and **metrics** obtained using the eye tracker device. In this case we can only comment on the data for the five participants whose behavior was recorded by the eye tracker. Most metrics calculated during an eye tracking session are calculated from *fixations*. The two metrics to determine the level of *interest* and therefore *usefulness* of a particular part of the interface are [7]: the *number of fixations on an AOI* (**#Fij**) and *the percentage of total inspection time spent looking a certain AOI* (**%Insp**). The calculated durations and times are measured in seconds. In Table 6 we can see the values of all these metrics for the six areas of interest defined in the COLLECE user interface.

Table 6.
Metrics provided by the eye tracker device**

Area of Interest (AOI)-awareness	Number of fixations per AOI (#Fij)*	Percentage of time spent on AOI (%Insp)*
AOI-Session Panel	**26.00 (17.82)**	**0.52 (0.31)**
AOI-Editor	16.20 (10.43)	0.41 (0.30)
AOI-Multi-user scrollbar	13.20 (13.66)	0.35 (0.38)
AOI-Semaphores	11.40 (8.17)	0.16 (0.10)
AOI-Global state	5.60 (2.41)	0.10 (0.06)
AOI-Code Line	**2.00 (2.35)**	**0.03 (0.03)**

* We show the mean and, in parentheses, the standard deviation.
** Values for *n*=5 participants (whose behavior was registered by the *eye tracker*).
Source: The authors.

Table 7.
Ratings of the *awareness* elements of COLLECE (TAM *framework*)**

Area of Interest (AOI)	Perceived Usefulness (PU) - Pretest*	Intention To Use (ITU) - Pretest*	Perceived Usefulness (PU) - Posttest*
AOI-Session Panel	**4.80 (0.45)**	**4.80 (0.45)**	**4.60 (0.89)**
AOI-Editor	4.00 (1.22)	4.00 (1.22)	4.00 (1.00)
AOI-Multi-user scrollbar	3.60 (1.67)	3.60 (1.67)	3.80 (1.79)
AOI-Semaphores	**4.80 (0.45)**	4.60 (0.55)	**4.60 (0.55)**
AOI-Global state	4.00 (1.00)	4.00 (1.00)	3.00 (1.00)
AOI-Code Line	4.40 (0.89)	4.20 (0.84)	3.60 (1.95)

* We show the mean and, in parentheses, the standard deviation.
** Values for *n*=5 participants (whose behavior was registered by the *eye tracker*).
Source: The authors.

Considering the values calculated for each AOI (Table 6), we see that the most consulted element is the session panel (greater number of fixations and percentage of time inspecting this element). These metrics coincide with the participants' subjective assessment in the TAM questionnaire (Table 7), since it was considered to be the most useful element (PU) and with more intention to use (ITU) (M = 4.80). According to calculated metrics, the elements which were consulted less by the participants (which were dedicated less inspection time) were the indicator of the global state and the line number on which the other member of the group is working (Table 7). Indeed, in the posttest these were the least valued elements (the valuation of PU decreases for both elements in relation to the rating given to them in the pretest) (Table 7).

Finally, as discussed in Section 4.5, the students were asked to rate the support provided by the COLLECE user interface to each of the dimensions included in the **framework of Gutwin and Greenberg** [22]. In Table 8 we can see the values assigned to each of them. Participants considered the support to dimension "*who"* of COLLECE was very good, mainly emphasizing the assessment of the sub-dimension *identity* (M = 4.70). This information is displayed in the session panel of the application, which, as has been shown, is the most consulted element (highest %Insp) and considered most useful (high value of PU) by participants in the activity. As for the dimension "*what"*, that defines the action being done by other group members and what objects are being manipulated, was assessed positively, although not as much as the previous dimension. As for the sub-dimensions relating to "*what"*, those sub-dimensions that refer to *changes* being made by members of the group (M = 4.30) and what *artifacts* were performed (M = 4.00) were the most valued. The one that indicates the *level of activity* (M = 2.80) was considered as less supported by the COLLECE interface. The dimension "*where"* was considered the worst supported by the system. In fact, the sub-dimensions that were scored the worst were those that refer to where the other group members are looking. That dimension is not supported by COLLECE (nor by most existing groupware systems), since eye tracking equipment would be necessary to record such information and transmit it to other group members. This is the idea behind recent works dealing with *gaze transfer* in the context of groupware systems and even in contexts of collaborative programming [28,29].

After completion of the test phase and posttest, students watched the recordings made during the experimental task, thus performing a **retrospective thinking aloud** (RTA) session, which sought to ascertain the impressions of the participants regarding the COLLECE system. A brief **interview** was then conducted with participants. This phase allowed for the identification of which were considered to be the strengths and weaknesses of the system and possible suggestions for improvements. Among the elements that were most valued by the participants is the support to coordination based on the assignment of turns and, among the most criticized, the *structured chat*. Also several participants suggested improving the identification of *where* the other member of the group is editing. Although this information is visually shown by the "AOI-Code Line", perhaps its location at the interface, its visualization format or size is not ideal, since, as has been found, despite being one of the elements that was better valued in intention to use (ITU) in the pretest, it was less focused onby students (%Insp) and considered to be less useful for the task (PU in posttest). Participants also proposed including an additional audio channel, to display the numbering of the lines of code in the shared context (in the source code) and to increase the flexibility of the decision-making policy, which in the current version of COLLECE is based on consensus of all group members. This latest enhancement will be of particular interest when the number of members of each group is greater than two.

Table 8.
COLLECE's support given to the awareness dimensions and sub-dimensions of the Gutwin and Greenberg framework .*

Dimension	Sub-dimension	Score*
Who	Presence	**4.50 (0.85)**
	Identity	**4.70 (0.67)**
	Authorship	**4.50 (0.71)**
What	Action	3.60 (1.07)
	Intention	3.30 (1.16)
	Artifact	**4.00 (0.94)**
	Changes	**4.30 (0.95)**
	Activity Level	2.80 (1.14)
Where	Location	3.50 (1.43)
	Gaze	**2.90 (1.20)**
	View	**2.90 (1.20)**
	Reach	3.30 (1.42)

* We show the mean and, in parentheses, the standard deviation.
** Values for *n*=10 participants.
Source: The authors.

5. Concluding remarks and future works

In this work we have evaluated COLLECE, a system that provides support to distributed collaborative programming. Unlike previous evaluations, the work carried out combines several techniques for the evaluation of multiuser interactive systems. Thus, we have comprehensively applied the following techiques: inspection (heuristic evaluation), subjective (questionnaires, interviews) and objective (automatic logging) inquiry, as well as testing in a usability lab (RTA, eye tracking and recording of the use). While checklist-based heuristic evaluation is a technique usually found in the literature [21], the use of eye tracking techniques applied to the evaluation of awareness support is a more original.

We are aware of the small sample size in this first study, so the results obtained should be considered as preliminary. The size is mostly due to the difficulty of analyzing dynamic information collected when applying techniques of eye tracking. Therefore, as a continuation of this work, we are considering replicating this experiment with a bigger sample, in order to obtain some more concluding results. Another line of continuation we expect to address is the possibility of incorporating the technique of *gaze transfer* in a collaborative system and testing its use as a technique for awareness support [24]. This is feasible, as we own the hardware that may be necessary in order to carry it out.

Acknowledgements

This work has been partially supported by the coordinated project EDUCA-Prog, of the Ministerio de Ciencia e Innovación (Spain) (TIN2011-29542-C02-02), the CYTED Project (Net 513RT0481) and the Government of Castilla-La Mancha's (JCCM) Gite-Learn Project (PEII-2014-012A) . The authors also want to thank the UCLM students who took part in the experience for their collaboration.

References

[1] Greif, I., Computer-supported cooperative work: A Book of Readings. San Mateo, CA: Morgan Kaufmann, 1988.

[2] Bravo, C., Duque, R. and Gallardo, J., A groupware system to support collaborative programming: Design and experiences. Journal of Systems and Software, 86 (7), pp. 1759-1771, 2013. DOI: 10.1016/j.jss.2012.08.039

[3] Bravo, C., Redondo, M.A., Verdejo, M.F. and Ortega, M., Framework for process and solution analysis in synchronous collaborative learning environments. International Journal of Human-Computer Studies, 66 (11), pp. 812-832, 2008. DOI: 10.1016/j.ijhcs.2008.08.003

[4] Dourish, P. and Bellotti, V., Awareness and coordination in shared workspaces, Proceedings of the Conference on Computer Supported Cooperative Work (CSCW), pp. 107-114, 1992. DOI: 10.1145/143457.143468

[5] Gallardo, J., Molina A.I., Bravo, C., Redondo, M.A. and Collazos, C.A., An ontological conceptualization approach for awareness in domain-independent collaborative modeling systems: Application to a model-driven development method. Expert Systems with Applications, 38 (2), pp. 1099-1118, 2011. DOI: 10.1016/j.eswa.2010.05.005

[6] Nielsen, J. and Pernice, K., Técnicas de eye tracking para usabilidad Web. ANAYA Multimedia, 2010.

[7] Poole, A. and Linden, J.B., Eye tracking in human-computer interaction and usability research: Current status and future prospects Pennsylvania: Idea Group, Inc, 2005.

[8] Carroll, J.M., Neale, D.C., Isenhour, P.L., Rosson, M.B. and McCrickard, S.D., Notification and awareness: synchronizing task-oriented collaborative activity. International Journal of Human–Computer Studies, 58 (5), pp. 605-632, 2003. DOI: 10.1016/S1071-5819(03)00024-7

[9] Stefik, M., Bobrow, D.G., Foster, G., Lanning, S. and Tatar, D., WYSIWIS revised: Early experiences with multiuser interfaces. ACM Transactions on Office Information Systems, 5 (2), pp.147-167, 1987. DOI: 10.1145/27636.28056

[10] Gutwin, C., Roseman, M. and Greenberg, S., A usability study of awareness widgets in a shared workspace groupware system., Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW), pp. 258-267, 1996. DOI: 10.1145/240080.240298

[11] Gallardo, J., Molina, A.I. and Bravo, C., A framework for the design of awareness support in collaborative situations of implicit interaction, Proceedings of the 13th International Conference on Interacción Persona-Ordenador - INTERACCION, 2012. DOI: 10.1145/2379636.2379643

[12] Johnson, P.M., Reengineering inspection: The future of formal technical review. Communications of the ACM, 41 (2), pp. 49-52. 1998. DOI: 10.1145/269012.269020

[13] Shen, H. and Sun, C., RECIPE: A prototype for Internet-based real-time collaborative programming, Proceedings of the 2nd Annual International Workshop on Collaborative Editing Systems, 2000.

[14] Jo, C.H. and Arnold, A.J., A portable and collaborative distributed programming environment, International Conference on Software Engineering, pp. 198-203, 2003.

[15] Devide, J., Meneely, A., Ho, C-W, Williams, L. and Devetisikiotis, M., Jazz sangam: A real-time tool for distributed pair programming of a team development platform, Proceedings of Infrastructure for Research on Collaborative Software Engineering (IReCoSE), 2008.

[16] Rubin, J. and Chisnell, D., Handbook of usability testing. How to plan, design and conduct effective tests. Indianapolis, Indiana: Wiley Publishing, Inc., 2008.

[17] Pinelle, D. and Gutwin, C., Groupware walkthrough: Adding context to groupware usability evaluation, Proceedings of the 2002 SIGCHI Conference on Human Factors in Computing Systems, pp. 455-462, 2002. DOI: 10.1145/503376.503458

[18] Baker, K., Greenberg, S. and Gutwin, C., Heuristic evaluation of groupware based on the mechanics of collaboration, Proceedings of the 8th IFIP working conference on engineering for human-computer interaction (EHCI), pp 123-140, 2001. DOI: 10.1007/3-540-45348-2_14

[19] Convertino, G., Neale, D., Hobby, L., Carrollo, J. and Rosson, M., A laboratory method for studying activity awareness, Proceedings of the Third Nordic Conference on Human-Computer Interaction, pp. 313-322, 2004. DOI: 10.1145/1028014.1028063

[20] Röcker, C. and Magerkurth, C., Privacy and interruptions in team awareness systems. Universal acess in human computer interaction. Coping with Diversity Lecture Notes in Computer Science Volume 4554, pp 273-283, 2007.

[21] Antunes, P., Herskovic, V., Ochoa, S.F. and Pino, J.A., Reviewing the quality of awareness support in collaborative applications. Journal of Systems and Software, 84, pp. 146-169, 2014. DOI: 10.1016/j.jss.2013.11.1078

[22] Gutwin, C. and Greenberg, S., A descriptive framework of workspace awareness for real-time groupware. The Journal of Collaborative Computing, 11 (3-4), pp. 411-446, 2002. DOI: 10.1023/A:1021271517844

[23] Hennessey, C., Framework for colocated synchronous dual eye tracking. Proceedings of the Dual Eye-Tracking in CSCW (DUET), 2012.

[24] Mueller, R., Helmert, J.R., Pannasch, S. and Velichkovsky, B.M., Following closely? The effects of viewing conditions on gaze versus mouse transfer in remote cooperation. Proceedings of the Dual Eye-Tracking in CSCW (DUET), 2011.

[25] Nielsen, J. and Pernice, K., Eyetracking methodology. How to conduct and evaluate usability studies using eyetracking. Nielsen Norman Group, 2009.
[26] Davis, F.D., User acceptance of information technology: System characteristics, user perceptions and behavioral impacts. International Journal of Man–Machine Studies, 38 (3), pp. 475-487, 1993. DOI: 10.1006/imms.1993.1022
[27] Legris, P., Ingham, J. and Collerette, P., Why do people use information technology? A critical review of the technology acceptance model. Information and Management, 40 (3), pp. 191-204, 2003. DOI: 10.1016/S0378-7206(01)00143-4
[28] Bednarik, R. and Shipilov, A., Gaze cursor during distant collaborative programming: A preliminary analysis, Proceedings of the Dual Eye-Tracking in CSCW (DUET), 2011.
[29] Bednarik, R., Shipilov, A. and Pietinen, S., Bidirectional gaze in remote computer mediated collaboration: Setup and initial results from pair-programming, Proceedings of the ACM 2011 conference on Computer supported cooperative work, pp. 597-600, 2011. DOI: 10.1145/1958824.1958923

AI. Molina, received her degree in Computer Science in 2002 and her PhD. in 2007 from the University of Castilla–La Mancha, Spain. She has since joined the Escuela Superiorde Informática (College of Computer Science and Engineering) at the University of Castilla–La Mancha, Spain. In addition to teaching, her main interests are in the field of new information technologies applied to collaborative learning and computer-human interaction.

J.Gallardo, received his BSc., MSc. and PhD. from the Universidad de Castilla-La Mancha, Spain. He has been an assistant professor at that University since 2011, and is now a professor at the Universidad de Zaragoza, Spain. He has been a member of the CHICO research group at the Universidad de Castilla-La Mancha, Spain, since 2006. His research interests include CSCW, groupware development and the application of model-driven engineering to such fields.

M.A. Redondo, has a PhD. in Computer Science in 2002 and is an associate professor at the Escuela Superior de Informática (College of Computer Science and Engineering) at the University of Castilla-La Mancha, Spain. His research interests focus on the fields of new Information Technologies applied to Computers in Education and Computer-Human Interaction.

C. Bravo, received his MSc. in Computer Science in 1996, from the Universidad de Sevilla, Spain and his PhD. in 2002 from the Universidad de Castilla–La Mancha, Spain. He joined the Computer Science Engineering Faculty of the Universidad de Castilla–La Mancha, Spain in 1998. His research interests include computer-support for collaborative learning, model-driven approaches to groupware engineering, and collaborative programming.