The Journal of Systems and Software 220 (2025) 112276

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Awareness support in collaborative programming tools: An evaluation based on programmer's perception and eye tracking

Ana I. Molina [a,*], Crescencio Bravo [a], Jesús Gallardo [b], Carmen Lacave [a], Miguel A. Redondo [a]

[a] Department of Information Technologies and Systems, University of Castilla-La Mancha, Spain
[b] Department of Computing and Systems Engineering, Aragon Institute for Engineering Research (I3A), University of Zaragoza, Spain

ARTICLE INFO

Editor: Dr. Nicole Novielli

Keywords:
Collaborative programming
Awareness
Groupware
Eye tracking
Software evaluation

ABSTRACT

Groupware technology is an essential asset for organizations and a successful medium to support social meetings and teacher-learning processes when people are geographically distributed. Computer programming is a domain that can take advantage of the Collaborative Work paradigm and of this technology. However, to be truly effective, groupware systems must provide suitable *awareness* support, i.e., be capable of informing users of the activity taking place and of the team outcome, so that they can build a setting for their work. The study of awareness mechanisms and their influence on collaborative programming processes is an open research question. In this context, this article contributes an experimental study to evaluate the mechanisms to support coordination, communication, and awareness of COLLECE, a distributed group programming system rich in awareness support. This study brings as a novel approach the combination of subjective and objective information sources, specifically eye tracking techniques, and incorporates a heuristic evaluation of awareness support based on a well-known framework. Thanks to this experiment, we have been able to verify that this blended approach, in which users participate intensively, provides a more comprehensive and deeper assessment of awareness and other coordination and communication mechanisms, and thus a better understanding of how their use can influence the collaborative programming process and outcome (program or piece of software). The study has revealed strengths, weaknesses and opportunities for improvement of the evaluated system, and has yielded concrete results as to which are the most useful awareness components of the user interface not related to the activity supported by the system (programming), these being the traffic lights; the easiest to use, such as the session panel; or the components that impose a greater cognitive load, such as the multi-scrollbar. Regarding the relationship between the supported awareness dimensions and the quality of the collaboration outcome (computer program), the results point to the need to adequately support the *What-Artifact* dimension, i.e., highlighting in which part of the shared workspace (the source code) the user with floor control is working.

1. Introduction

Nowadays, collaborative systems, or groupware, are an essential asset for any organization and an excellent medium for social and family encounters between people who are distant from each other (Ciolfi et al., 2023), as well as for supporting learning processes, as evidenced by the recent COVID-19 pandemic (Pokhrel and Chhetri, 2021).

Groupware systems, subject of study in the CSCW (Computer-Supported Cooperative Work) research area,[1] are most effective when they have an interface capable of informing users of the activity taking place and the outcome of teamwork, especially when the work is performed remotely (Collazos et al., 2019). When the face-to-face scenario, considered quite effective, must be recreated in a distributed way, knowledge elements are needed to provide rich and subtle additional information that contributes to better monitoring of group work (Schmidt and Randall, 2016). It is in this context that the concept of *awareness* arises. Awareness is defined by Dourish and Bellotti as "*knowledge of the activities of other users, which provides a context for one's*

* Corresponding author.
E-mail address: AnaIsabel.Molina@uclm.es (A.I. Molina).

[1] CSCW (Computer-Supported Cooperative Work) is an interdisciplinary field that studies how technology can facilitate collaborative work, encompassing both technical and social aspects. Introduced by Irene Greif and Paul M. Cashman in 1984, CSCW explores how computer systems can support group activities and coordination across time and space, focusing on both synchronous and asynchronous collaboration (Greif, 1988). It includes the design of collaborative systems like groupware and examines their social and organizational implications, aiming to improve both productivity and interpersonal interaction (Grudin, 1994).

https://doi.org/10.1016/j.jss.2024.112276
Received 25 July 2023; Received in revised form 7 September 2024; Accepted 31 October 2024
Available online 1 November 2024

own activity" (Dourish and Bellotti, 1992). This information makes it possible to know, but also to anticipate, the actions of others, resulting in more effective collaborative interaction (Gross, 2013). Providing adequate support for awareness by including communication and coordination mechanisms in the user interfaces to provide users with perception and understanding of the activity of others involved in the group activity has been a research challenge since the emergence of groupware systems (Dourish and Bellotti, 1992; Schmidt, 2002) and continues to be so today (Collazos et al., 2019; Mantau and Benitti, 2022b, 2024; Richter et al., 2024). Particularly, the study of awareness mechanisms and their influence on collaborative processes is an open research question (Duckert and Bjørn, 2024; Mantau and Benitti, 2022a).

One domain in which the use of collaborative systems has been recurrently exploited is computer programming, giving rise to a scenario in which a group of programmers code together, possibly remotely, and test their programs (Chorfi et al., 2022; Lacave and Molina, 2021). The benefits of group programming have been demonstrated, both in the professional and educational areas (Alsaqqa et al., 2020; Beasley and Johnson, 2022; Xu and Correia, 2023), resulting in higher-quality products and improving the motivation and engagement of programmers (Celepkolu and Boyer, 2018; Hawlitschek et al., 2022), who obtain positive effects both at the technical level and in terms of soft skills (negotiation, decision-making, and ability to share best practices, among others) (Adeliyi et al., 2021; Tsai et al., 2023).

A review of the literature on *distributed pair programming*[2] (da Silva Estácio and Prikladnicki, 2015) identified the lack of empirical studies that analyze the use of software systems that follow this paradigm and, in particular, the support they provide for communication and coordination. A more recent literature review focused on the use of the collaborative approach in the teaching of programming (Silva et al., 2020) also highlights the need for evaluation methods to identify which characteristics of group programming activities and of the tools that support them are most effective. Furthermore, its authors propose to analyze the group programming process more deeply, *combining quantitative and qualitative methods*.

Therefore, taking into account the importance of awareness support in collaborative systems in general and in programming in particular, as well as its influence on the process and outcome of collaborative programming activity, it is necessary to identify which communication, coordination, and awareness mechanisms should be included in group programming support systems for being the most useful for programmers and improving the working process and its outcome. Therefore, the motivation of this work is to provide empirical knowledge and experience to address this research challenge and to seek to determine which collaboration support mechanisms should be included in group programming support systems. In particular, this paper contributes with a study that analyzes the collaboration and awareness support mechanisms in a group programming system. The system chosen for that is COLLECE (Bravo et al., 2013), developed by some of the authors of this paper and used as a prototype for research and programming learning, and a good representative of this system category, as will be explained below. For this study, a combination of several assessment methods (heuristic evaluation, subjective perception questionnaires, testing and automatic logging, and eye tracking) is proposed for obtaining quantitative and qualitative measures.

One of the most novel aspects of this experimentation is the incorporation of the eye tracking technique. The use of this technology is especially useful for understanding the underlying cognitive processes of the participants and, in particular, aspects related to attention or mental effort (Bojko, 2013; Endres et al., 2023). Although this technique has been used in the field of software engineering to analyze model and source code comprehension as well as debugging and traceability processes, there are few works that have used it to study group programming (Obaidellah et al., 2018; Sharafi et al., 2015b), and even less that address the support for awareness in this programming paradigm (Molina et al., 2015).

Therefore, the objective of this research work is twofold. On the one hand, it aims to research how the communication, coordination, and awareness support mechanisms included in a group programming system can influence the programming task. Specifically, it aims to determine which mechanisms are more useful, generate less cognitive load, and improve the joint programming process in order to take them into account in the design or redesign of this type of system. On the other hand, a detailed description of the design and execution of an evaluation experiment is provided for its possible replication to assess such support in other collaborative programming systems. This evaluation method brings as a novel approach the combination of subjective and objective information sources, specifically eye tracking techniques, and incorporates awareness-support evaluation frameworks well consolidated in the literature, such as those of Gutwin and Greenberg (Gutwin and Greenberg, 2002).

The next section presents the background of this research, discussing both the main works related to awareness support in collaborative systems and their assessment, the case of group programming systems and how they support this feature, as well as a brief description of the COLLECE programming system. Section 3 describes the empirical evaluation of this groupware application: the research questions formulated, the equipment and measures used, the evaluation carried out based on data collection, and the statistical analyses applied. Then, Section 4 presents the results obtained and discusses how they allow answering the research questions posed, discusses implications for practice and transferability, and the validity of the evaluation. Finally, Section 5 summarizes and concludes the article and states the future work derived from it.

2. Background

This section reviews the concept of awareness, how it is evaluated through different and complementary techniques, and how it is supported in different collaborative programming systems.

2.1. Awareness support in groupware

Awareness is a concept that must be considered when dealing with collaborative systems (Mantau and Benitti, 2022b). It refers to how users get informed about who else is working in the system and which work is being carried out (Dourish and Bellotti, 1992). Traditionally, awareness support in collaborative systems has been categorized into the following four types (Gutwin et al., 1996; Gutwin and Greenberg, 2002): *informal awareness,* which is about the perception of who is there and what he/she intends to do; *social awareness*, which refers to information about other users in a social context; *group structure awareness,* which is about the roles, responsibilities, and status of the other members; and *workspace awareness,* which implies knowledge about the interactions of others with a shared workspace and the artifacts on it.

Other authors have suggested other taxonomies for awareness types. This way, Antunes et al. (2014) have proposed a different awareness organization: *collaboration awareness*, referring to the perception of the availability of the group by the participants; *location awareness*, which is about perceiving where another member is located, oriented, moving towards, and looking at; *context awareness*, referring to what is happening in the shared space; *social awareness* and *workspace awareness*, which are both present in Gutwin and Greenberg's work (2002); and *situation awareness*, which is about being aware of the different situations happening around you, knowing why things are happening, and thinking of an action to tackle those (Mitaritonna et al., 2019).

[2] *Pair programming* refers to the orchestration protocol between collaborators in which a *driver* writes code and one or more *observers or navigators* review this code in real time (in turn-taking or an agreed approach).

Lastly, other types of awareness information that are useful in CSCW systems are *activity awareness, availability awareness, perspective awareness, presence awareness,* and *rhythm awareness* (Collazos et al., 2019). Whereas other kinds of groupware may need different types of awareness, such as CSCL (Computer-Supported Collaborative Learning) systems, which may need shared knowledge awareness (Collazos et al., 2003; Schnaubert and Bodemer, 2022).

Support for all these awareness dimensions involves a specific design of the user-computer and user-user interaction and the inclusion of certain components and widgets in the user interfaces of groupware applications to provide information to those involved in the collaborative task about the identity, activity, and location of the other group members. The corresponding awareness mechanisms are based on the use of identifying avatars, maps and radars, color coding, multi-scroll bars, and telepointers, among others (Gutwin et al., 1996). This classic set of components has been enriched in recent years with the incorporation of new sources of information, based on the use of specialized sensors, visual and auditory signals, etc., necessary in the new interaction contexts that have been emerging (Gallardo et al., 2018; Lopez and Guerrero, 2017; Mantau and Benitti, 2022a).

Other research efforts target specific types of awareness. For example, in Mitaritonna et al. (2019), the authors define a model for situational and workspace awareness, focusing mainly on the former. This framework is aimed at systems where augmented reality plays an important role. Other authors are focusing on supporting so-called *emotional awareness* (Collazos et al., 2021), whose use can influence learners' motivation, engagement, and self-regulation, mainly in collaborative learning environments (Arguedas et al., 2016). A very comprehensive and up-to-date systematic review of the different types of awareness can be found in Mantau and Benitti (2024).

2.2. Methods for assessing awareness support in collaborative systems

The evaluation of collaborative systems and awareness support remains a research challenge (Mantau and Benitti, 2022b). Although there are well-established and widely used methods for usability evaluation of single-user systems, such as heuristics and cognitive walkthroughs, and techniques, such as thinking aloud (Stone et al., 2005), they are not easy to adapt and apply when evaluating groupware systems (Collazos et al., 2019). Thus, the so-called groupware usability is 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*" (Gutwin and Greenberg, 2000). For this reason, specific techniques have been proposed for the evaluation of groupware systems (Frías et al., 2019; Pinelle and Gutwin, 2000b, 2000a). Some are adaptations of well-known techniques in the field of Human-Computer Interaction (HCI) such as heuristic evaluations (Baker et al., 2001; Gutwin and Greenberg, 2002) or cognitive walkthroughs (Pinelle and Gutwin, 2002), which have been tailored to support groupware evaluation.

The workspace awareness framework by Gutwin and Greenberg (2002) is one of the most widely used works about the verification of awareness support in synchronous collaborative systems. This framework entails a heuristic evaluation made up of ten questions that must be answered to study whether adequate support for workspace awareness is being provided. This work focuses on classic scenarios where users deal with a graphical user interface so that they handle a shared context (a set of objects that are viewed and manipulated jointly by the participants of a group work activity) (Ellis et al., 1991).

Also, another interesting approach along the line of heuristics for awareness support evaluation is an adaptation of the well-known usability heuristics by Nielsen and Molich (1990) for their application to groupware systems (Baker et al., 2001). The authors started with the *Mechanics of collaboration* framework and set it out as a set of heuristics for identifying problems in visual shared work surfaces. More recently, a set of design guidelines or heuristics that can also be used for collaborative system evaluation was proposed by Cepero et al. (2021).

A similar, yet different, kind of approach for evaluating awareness support is the use of checklists. Here, it is worth mentioning the work by Antunes et al. (2014). They identified 54 design elements, grouped into 14 categories, and later specified a question for each design element, creating a checklist for the evaluation of awareness support through the six types of awareness the authors tackled. Some authors have made adaptations and simplifications of this awareness checklist (Do Espirito Santo et al., 2018).

The use of questionnaires to measure the subjective perception of participants in collaborative activities supported by groupware systems is also a subject of interest in this field, with recent contributions such as the creation of the SWUS (*Shared Workspace Usability Scale*) instrument (Berkman et al., 2018) and of the *Awareness Assessment Model* (Mantau and Benitti, 2024, 2023). The combination of evaluation methods (questionnaires, on-screen behaviour recordings, and interviews) is common in this field in order to perform a more complete analysis of the collaborative process (Geszten et al., 2021, 2024).

Finally, and in addition to the aforementioned frameworks (which are more theoretical or that use more subjective sources of information), a different and promising line of work proposes the use of eye tracking techniques for the evaluation of awareness support in collaborative systems (Molina et al., 2015). The usefulness of this technique lies in the *eye-mind assumption*, which argues that there is a link between the visual scanning behaviour, and the cognitive activity, so that we pay attention to that part of the stimulus that we are processing at each moment when using a computer system (Just and Carpenter, 1980). Although this relationship is not always true, as we do not always think about or pay much attention to what we are viewing, it is sufficiently consistent to draw objective conclusions about the cognitive processes that cause the *fixations*[3] and visual scanning behaviour of the subjects (Bojko, 2013). In relation to this aspect, Poole and Ball (2006) present a compilation of the main metrics used in HCI research and their interpretation. Thus, for example, a gaze of longer duration on an image area generally indicates an increased difficulty in interpreting its content, or a greater number of fixations on a particular area may indicate an increased interest by the user in its content.

This objective evaluation technique has been extensively used for the evaluation of interactive systems (Bojko, 2013), mostly web pages (Nielsen and Pernice, 2010; Wang et al., 2014), and multimedia learning resources (Coskun and Cagiltay, 2022), as well as for analysing the visual behaviour of programmers or programming apprentices (Andrzejewska and Kotoniak, 2020; Obaidellah et al., 2018; Sharafi et al., 2015b). Thus, for example, its use has allowed to analyze the visual behaviour of the programmer when trying to read (Turenko et al., 2019), comprehend (Andrzejewska and Skawińska, 2020; Peitek et al., 2020), debug (Katona et al., 2019) or detect errors (Li et al., 2019; Liu et al., 2020) in the code of a program, as well as identify different analysis patterns according to different programmer profiles such as, for example, their expertise (Andrzejewska and Kotoniak, 2020; Lin et al., 2016; Yenigalla et al., 2016), age (Papavlasopoulou et al., 2017) or gender (Huang et al., 2020; Hung and Wang, 2021; Obaidellah and Haek, 2018), among others. However, there are very few studies that use eye tracking to analyze group programming processes, and even fewer that evaluate awareness support in software tools to carry out collaborative programming (Obaidellah et al., 2018; Sharafi et al., 2015b; Villamor and Rodrigo, 2017).

A systematic literature review on the use of eye tracking in software engineering is presented in Sharafi et al. (2015b). This review concludes that the main uses of eye tracking have been to analyze model comprehension, code comprehension, debugging, collaborative interaction, and traceability. In the case of collaborative interaction, only three papers were found, and those that were available focused on using

[3] In eye tracking, *fixation* refers to a period of time during which the eye is relatively stable and focused on a particular point in the visual field.

this technique to record the visual behaviour of programmers working in groups and to make a posteriori analysis of the common areas of interest during the programming task (Jermann and Nüssli, 2012; Sharma et al., 2013). In other works, the information recorded by the eye tracker was used as cues for the areas where expert programmers were looking, which could help novice programmers understand a code or detect bugs (Stein and Brennan, 2004). These cues were not provided synchronously but were recorded to be consulted later. In this sense, the hypothesis of these authors was that a novice programmer could benefit from previewing the experts' eyegaze patterns before performing the bug-finding task.

In Obaidellah et al. (2018), a systematic review focused specifically on the use of this technique in computer programming is presented. This review shows that most of the works found apply eye tracking in program comprehension and debugging tasks, with only a few works on collaborative programming and requirements traceability. Thus, only five works on the use of eye tracking in group programming scenarios are identified. Some ones use it to analyze shared visual attention during pair programming, but in non-distributed work scenarios (Pietinen et al., 2008, 2010), for which there are two eye trackers in the same workspace. In other cases, it is used in distributed scenarios, making a dual tracking with the aim of directing the attention of the partner of the pair while performing analysis tasks of the same code (Jermann and Nüssli, 2012), i.e., to support the so-called "gaze awareness" as an implicit coordination mechanism in distributed collaborative programming scenarios (D'Angelo and Begel, 2017). In the same line of work, Villamor and Rodrigo (2017) describe a dual eye tracking study in which barely 16 pairs of students participated and in which the aim was to find out the patterns of code analysis in tracing and debugging tasks in pairs; the students were located in a distributed manner and could only communicate via chat. The objective in this case was to analyze the existing differences when the pairs were formed by individuals with the same or different levels of prior knowledge of the tasks to be performed. The results indicated that mixed pairs presented greater difficulties in coordinating.

Therefore, we conclude that there are no studies, apart from a previous work by the authors of this research (Molina et al., 2015), that have used the eye tracking technique for the objective addressed in this work, that is, to evaluate the support for awareness and specifically its usefulness, the cognitive load associated with its use, as well as its influence on group programming tasks. To our understanding, there is great potential in using this new source of information for such a purpose since the measures provided by the eye tracker allow one to contrast and complement the other sources of information (of a more subjective nature) (Endres et al., 2023; Obaidellah et al., 2018; Sharafi et al., 2015b). The eye tracker provides objective, quantitative, and real-time data about the elements that are being attended to during the execution of the programming task (Endres et al., 2023). Its use complements traditional methods, whose results may be subject to bias. Such biases do not necessarily have to be intentional, but it has been shown that there is not always a coincidence between what users perceive about their own thoughts and behaviors and the underlying processes or intentions (Carter and Luke, 2020).

2.3. Awareness support in collaborative programming IDEs

IDEs (*Integrated Development Environments*) are applications designed to offer programming and development functionalities to software builders. They commonly include a source code editor, a compiler, and a virtual machine to run the code (for one or more programming languages) as basic components, and additionally, they can contain a debugger and tools for browsing the project and its components. Apart from these features, these environments are evolving to support integrations with code repositories, deployment tools, command-line shells, and collaboration capabilities. In fact, there are now many groupware systems that allow groups of users to carry out programming tasks collaboratively and from different locations.

Table 1 collects a list of representative collaborative programming systems that have more recently appeared and are in use, including their nature and several features. They were selected through a non-systematic review of the literature and of popular publications in the field of software development and were analyzed by installing and using them to perform some test programming tasks or by studying their user guide. This scheme is useful for typifying and classifying collaborative programming systems, but more importantly, it tells us what kind of collaboration support and awareness components can be expected in this kind of system, which is what we are interested in for evaluating such support.

As for the ***purpose of the system***, there are systems built for professional and commercial uses aiming at productivity in software development, prototype systems for research purposes, and environments for teaching-learning programming, or a combination of these (Table 1). In the field of programming learning systems, the visualization and tracing of programs have been exploited for didactic purposes (Carlisle et al., 2005; Jiménez et al., 2022; Lacave et al., 2020), and there are even some developments that have exploited the use of advanced interaction styles and paradigms to support this type of task (Arroyo et al., 2018; Schez-Sobrino et al., 2021). However, most of these tools are not collaborative. Regarding collaborative programming learning, a systematic review (Revelo et al., 2018) identified a set of collaborative strategies or techniques specifically oriented to support it: Git-Hub, collaborative IDEs, pair programming, peer code evaluation, elements of component-based software engineering, and hackathon-type events. In a similar line, other authors (Hundhausen and Carter, 2014) have stressed the importance of measuring the ability of programming environments to promote social interactions and awareness when programming, especially in programming teaching environments. In fact, these authors introduce the term "social programming environments" to refer to environments that take these factors into account.

Most of the systems in Table 1 consist of an IDE for software development extended with collaborative features; on the other hand, there are advanced code editors with support for the collaborative construction of source code and specific applications or environments as well.

In terms of ***technological support***, the current trend is to virtualize the IDEs and access them in the cloud, using the web navigator as the user interface and maintaining the same advanced features of the desktop-based IDEs (Table 1). Two well-known desktop IDEs are Eclipse and Visual Studio.

Leaving apart the programming-related collaborative tasks, the ***communication and coordination support provided*** in these settings varies quite a lot and usually consists of chat tools, forums, audio and video conferencing, collaborative whiteboards, screen sharing, voting tools, floor control, and other specific collaborative tools (Table 1).

To provide ***awareness***, several user interface (UI) components or widgets are incorporated in the different systems, being the more common elements the user lists or panels, and the tele-cursors (Table 1). The difference for us between a list and a panel of users is that the latter includes additional information about the user, such as his or her picture, assigned colour or working state. To support code editing, a few systems can highlight the blocks of text selected by the editor users. The radar view is a particularly interesting feature for displaying thumbnail representations of a complete source code, although it is not a widely used feature. The group scrollbar allows users to know the exact place in the source code where other users are editing or thinking about. For the coordination of certain tasks, the use of semaphore-like visual signals, as done by a couple of the systems analyzed, can be effective. Finally, the editing areas blocked for individual coding work must be clearly visible.

All the awareness mechanisms compiled in Table 1, implemented by means of a collaborative widget, have a spatial materialisation on the screen, except for the audio-video channels and sound awareness cues —for example, COLLECE beeps when chat messages and coordination actions are submitted— which we leave aside in this work, so that the use

Table 1
Comparative of more significant collaborative programming systems. The table includes general features of each of these systems (name and purpose), technological aspects (type of platform and implementation technology), the collaboration tools they incorporate (to support the programming task, as well as communication, coordination and awareness) and, finally, what are considered to be the most outstanding features of each one.

Name of collaborative programming system	System's purpose	Type of platform	Implementation technology	Collaboration support provided			Special features incorporated
				Programming tools	Communication and coordination tools	Awareness mechanisms supported	
RIPPLE (Boyer et al., 2008)	- Programming learning	IDE	Eclipse plug-in	- Code editing - Execution	- Chat - Log analysis - Tutoring	–	- Activity analysis
SAROS (Prechelt and Beecher, 2011)	- Programming research	IDE	Eclipse plug-in	- Code editing	- Chat - Audio conference - Whiteboard - Screen sharing	- User panel - Tele-cursors - Group scrollbar - Package explorer	- Follow mode
CoEclipse (Fan and Sun, 2012)	- Programming	IDE	Eclipse plug-in	- Code editing	- Chat - Automatic code lock	- User panel - Region visualization	- Consistency maintenance
CloudStudio (Estler et al., 2013)	- Programming research	IDE	Cloud-based	- Code editing - Debugger - Execution	- Chat - Audio-video conference	- User panel - Region visualization	- Changes tracking - Configuration management
CodeAnywhere[1]	- Programming	IDE	Cloud-based	- Code editing - Execution - Shared terminal	- Chat	- Tele-cursors - Radar view	–
Cloud9[2]	- Programming	IDE	Cloud-based	- Code editing - Execution - Shared terminal	- Chat	- User panel - Tele-cursors - Selection visualization - Changes history	–
Visual Studio + Live Share[3]	- Programming	IDE	Visual Studio	- Code editing - Debugger - Execution - Shared terminal	- Chat	- User panel - Tele-cursors - Selection visualization - Following participants	–
CodeTogether Live[4]	- Programming	Cross-IDE support	Eclipse, IntelliJ and Visual Studio plug-in	- Code editing - Execution - Shared terminal	- Chat - Audio-video conference - Screen sharing	- User list - Tele-cursors	- Multi IDE
CodeBunk[5]	- Programmig learning	Code editor	Cloud-based	- Code editing - Execution	- Chat - Audio-video conference	- User panel - Changes history	- Interviewing of developers
CodePen[6]	- Programming	Code editor	Cloud-based	- Code editing - Execution (Pen view: HTML, CSS, JS)	- Chat	- User list - Tele-cursors	- Front-end design
COPPER (Natsu et al., 2003)	- Programming research	Application	–	- Pair programming	- Chat - Floor control - Document list - Audio conference	- User panel - Coordination semaphores - Radar view - Document presence list - Operation status bar	- Distributed pair programming
COLLECE (Bravo et al., 2013)	- Programming learning - Programming research	Application	Java application	- Pair programming - Execution	- Chat - Coordination tools	- User panel - Tele-cursor - Group scrollbar - Coordination semaphores - Group state	- Distributed pair programming - Compilation history
COLLECE 2.0 (Lacave et al., 2019)	- Programming learning - Programming research	IDE	Eclipse plug-in	- Code editing - Execution	- Chat	- User panel - Tele-cursors - Region visualization	–
COLE-Programming (Jurado et al., 2013)	- Programming learning	IDE	Eclipse plug-in	- Code editing - Execution	- Chat - Forum - Voting pool	- User panel	- Evaluation of solutions using Fuzzy logic

[1] https://codeanywhere.com/.
[2] https://aws.amazon.com/cloud9.
[3] https://visualstudio.microsoft.com/es/services/live-share/.
[4] https://www.codetogether.com/.
[5] https://codebunk.com/.
[6] https://codepen.io/.

of such widgets can be studied through eye tracking techniques.

2.4. COLLECE: A distributed group programming support system

Our research group (CHICO, *Computer-Human Interaction and Collaboration*) has been working for more than 20 years in the field of design and evaluation of interactive and collaborative systems in the field of programming learning (Ortega et al., 2019). As a result of this work, several software tools have been developed. One of them is COLLECE (*COLLaborative Edition, Compilation and Execution of programs*), a distributed group programming support system that allows shared editing, compilation, and execution of programs (Bravo et al., 2013). This system so far supports two programming languages (C and Java) and incorporates a very wide and rich set of communication, coordination, and awareness mechanisms. In COLLECE, as in other similar systems, users are grouped into working sessions in which they solve a programming problem. This problem-solving task may involve developing code from scratch or modifying existing code.

A screenshot of the main components of the COLLECE user interface can be seen in Fig. 1. The main area of the COLLECE user interface is the collaborative code editor (1), in which the program is jointly coded. This program is then compiled and executed, and the result is shown in the console window (2). To be able to edit the code, a coordination protocol is established that regulates the editing turn. This is achieved by means of the floor control panel or turn panel (3). This is a graphical component by means of which users can express their intention to take the code editing turn. When a user has made a request to take a turn, the others give their consent, affirmative or negative, by means of the buttons on this panel. When all users agree to give the turn to the user who requested it, the change takes place. When deciding whether to compile and execute the program being edited, a similar procedure is followed, but using two other panels (4 and 5) that serve to make the agreements as to when to compile and when to execute, respectively.

Apart from those already mentioned, other notable components of the COLLECE user interface are the chat (6) and the session panel (7). The chat is structured, which means that it allows to classify messages thanks to a set of predefined sentence openers (e.g., "*I think that…*", "*I see a mistake in …*", "*I miss a …*") (Lund et al., 1996). As for the session panel, it includes the name and photo (or avatar) of each of the users participating in the session, as well as their working status (editing, compiling, etc.). Both name and status are highlighted in a specific color which, as will be seen later, identifies that user in certain awareness support elements.

In addition to the coordination support (floor control panel, compilation panel, and execution panel) and the synchronous communication tool (structured chat), it is worth mentioning COLLECE's awareness support. As discussed before, awareness support is an essential issue to consider in the design of distributed software (Collazos et al., 2019; Mantau and Benitti, 2022b), and it is necessary to develop effective group work or learning (Schnaubert and Bodemer, 2022). COLLECE incorporates a wide range of widgets and awareness mechanisms, providing real-time information about people, their status, and their actions. In Fig. 1, the following awareness support elements are highlighted: (a) path of the file being edited; (b) line being edited, highlighted with the identifying color of the user who is doing it (tele-cursor); (c) collaborative scroll bar (multi-scroll bar), to indicate where the editing user is working, also using the color that identifies him/her; (d) line number being edited at a given moment; (e) a traffic light (semaphore) that turns green when another user has made a request that requires a response; (f) name of the user who is currently editing; (7) the session panel, which includes (g) the user who edits, compiles, or executes with the corresponding status; (h) the user to whom the interface belongs within a box; and (i) the current working status of the session.

COLLECE has been widely used and evaluated (Bravo et al., 2013). However, in most of these evaluations, the source of information used has been of a subjective nature, collected through questionnaires on the opinion of students or software developers. In this paper, we propose to focus the evaluation on the awareness support incorporated in this system, complementing the use of questionnaires with other sources of objective information, such as the metrics provided by an eye tracker device (Molina et al., 2015) and other indicators related to interaction and collaboration (Duque et al., 2015; Duque and Bravo, 2008). For the latter, it is necessary to mention that COLLECE incorporates a logging module, which records and computes some indicators about the group programming activity (process) and the obtained solution (product), such as the contributions made in the chat and their types, the compilation errors, or the interactions count with the coordination and decision-making components. These indicators are recorded per group and per individual.

3. Empirical study

As explained before, the aim of this research work is to attempt to determine which communication and awareness mechanisms should be included in group programming support systems because they are more useful for programmers, generate less cognitive load, and improve the joint programming process. To analyze these aspects, an experiment was designed using a distributed collaborative programming system, COLLECE, that incorporates a complete set of components to support communication, coordination, and awareness (see Section 2.4). Furthermore, this tool has been widely used and tested by both computer engineering students and professionals (Bravo et al., 2013).

3.1. Research questions

To address the objective posed in this work, we draw on giving emphasis to the users' participation in the evaluation and to his or her subjective perception, as well as on ideas from the quality in use of a software product (Salomón et al., 2019). So that it is proposed to analyze the acceptance and usability (Burney et al., 2017), in terms of *usefulness* and *ease of use* (Davis et al., 2024), of the elements that make up the user interface (UI) of collaborative programming systems and, in particular, the support given to communication, coordination, and awareness. But we also seek to focus on the users' mental effort or attention when working with collaborative UI, taking advantage of heuristic evaluation practices of groupware products through checklists of requirement fulfillment (Baker et al., 2001; Gutwin and Greenberg, 2002), and the use of physiological measures, such as eye tracking (Ayres et al., 2021; Ayres and Paas, 2012; Bojko, 2013), which provide more objective measures related to the above aspects. In addition, we are interested in knowing the influence that the inclusion of specific collaborative UI components can have on the collaborative process itself as well as on the *product* obtained, according to previous research on process-product analysis approaches (Bravo et al., 2008). The ultimate goal is to be able to make well-founded decisions on the design or redesign of this type of application.

Therefore, we formulated the following research questions (RQ):

RQ1. What are the most *useful* collaboration support mechanisms (communication, coordination, and awareness) for programmers in the context of distributed collaborative programming tasks in COLLECE?
RQ2. What are the *easiest* collaboration support mechanisms *to use* in the context of distributed collaborative programming tasks in COLLECE?
RQ3. Is there a relationship between using certain components of the COLLECE user interface and obtaining a better *product* (source code created as a result of the programming task, henceforth, program)?
RQ4. Is there any relationship between the support for the dimensions of Gutwin and Greenberg's framework (*Who, What,* and

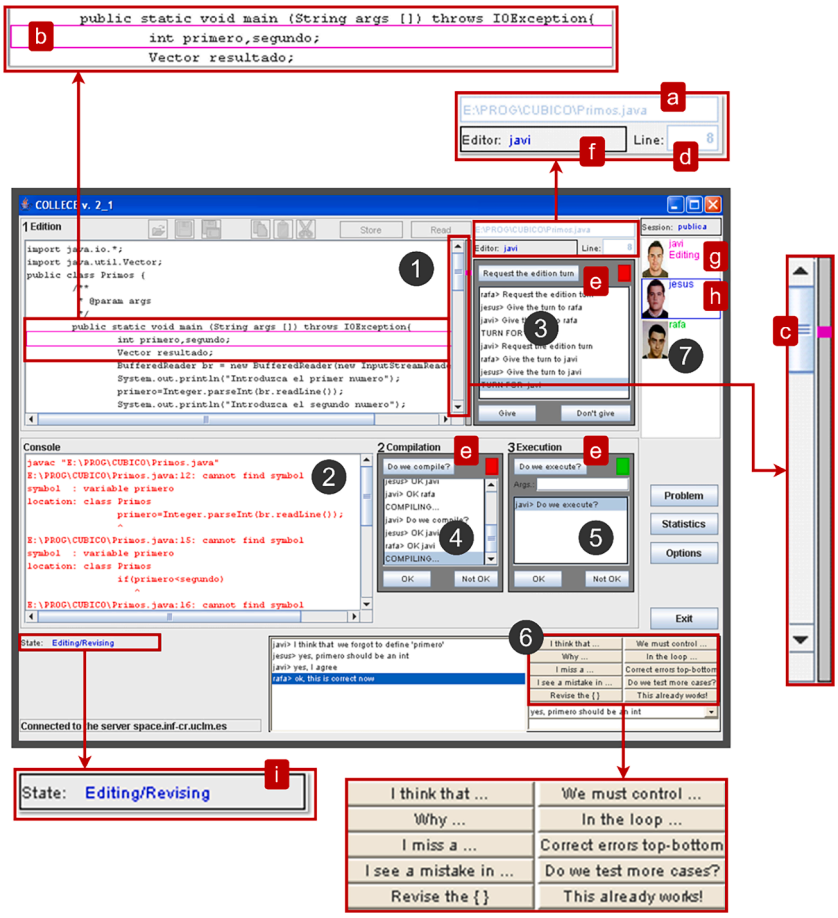

Fig. 1. Communication, coordination, and awareness mechanisms included in the COLLECE user interface. Main components in the interface: (1) Code editor; (2) Console; (3) Turn panel; (4) Voting panel (to decide to compile); (5) Voting panel (to decide to execute); (6) Structured chat and (7) Session panel. Awareness mechanisms: (a) Path of the file being edited; (b) Line being edited; (c) Multi-scroll bar; (d) Tele-cursor; (e) Semaphores; (f) User who is editing; (g) User who edits, compiles or executes with the corresponding status; (h) User to whom the interface belongs; (i) Current working status of the session.

Where) and the quality of the *product* (program) obtained collaboratively?

To answer all of them, an experiment was conducted in which end users programmed in pairs using the COLLECE system. This experiment combines several techniques commonly used for evaluating interactive systems (Stone et al., 2005): (1) *testing*, performed in a usability laboratory; (2) *inspection techniques* (heuristic evaluation, applying a well-known framework for evaluating awareness support, such as Gutwin and Greenberg's one); and (3) *inquiry methods*, by the use of subjective perception questionnaires and automatic logging techniques (to register the main interactions carried out in the system, such as chat contributions and interaction with some of its main components and widgets).

The next sections describe the details of the experiment conducted (Sections 3.2–3.7), in which communication, coordination, and awareness mechanisms in COLLECE are studied.

3.2. Equipment and settings

The experiment was performed in the Usability Lab of the CHICO (*Computer-Human Interaction and Collaboration*)[4] research group of the University of Castilla-La Mancha (UCLM). The laboratory includes the proper equipment for usability testing of interactive systems.[5] In this sense, the lab includes eye tracking testing equipment, several testing and interview zones (equipped with cameras and microphones), and an

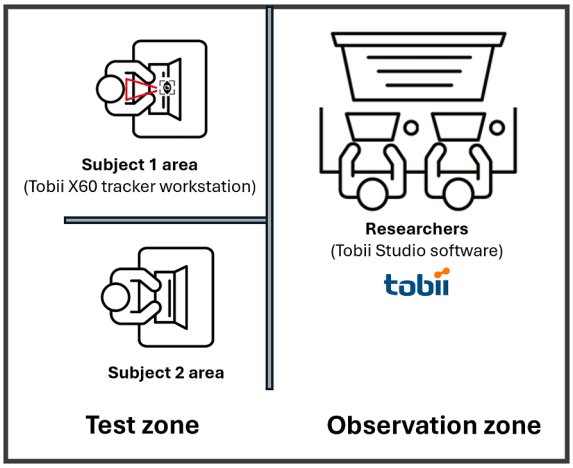

Fig. 2. Distribution of the zones in the usability laboratory.

[4] https://blog.uclm.es/grupochico/

[5] https://blog.uclm.es/grupochico/elementos/

observation zone for monitoring tests (Fig. 2). The equipment used for eye tracking is a Tobii X60 model, and the Tobii Studio software for the design, implementation, and subsequent analysis of eye tracking tests.

Since only one eye tracking device is available, in the development of the pair programming experiment, just the activity of one of the pair members was recorded with this device, while the activity of the pair partner was recorded by video camera. The interactions of both pair members were recorded using the system's logging module.

Previously, the set-up and adjustment of the physical space to be occupied by the subject to be recorded by the eye tracker device were necessary. These settings are essential when preparing an eye tracking session and include the adjustment of the distance, height (using a height-adjustable chair), and angle of the eye tracker device. Next, the calibration phase of the eye tracking device was performed. Calibration allows a personalized calibration profile to be created for each participant to ensure that the eye tracker accurately captures their eye movements. It is necessary to point out that some people are not trackable due to various reasons (by the shape of their eyes, eyelashes or eyelids, strabismus, or use of contact lenses or bifocals, among others).

It is necessary to comment that in both the design and development of the eye tracking session, the methodological recommendations of Nielsen and Pernice (2009) were followed, as well as the practical guide for eye tracking studies in software engineering proposed by Sharafi et. al. (2020).

3.3. Experiment design and development

Fig. 3 graphically illustrates the experimental design followed in this experiment.

Before carrying out the final experiment with the participants, a pilot test was carried out (involving three professors from the team of experimenters, playing the role of programmers) with the aim of setting up the laboratory. Specifically, tests were carried out to calibrate the eye tracker, control the lighting conditions in the room, and adjust the distances and angles between the seat and the screen. Some details regarding the materials and the protocol to be followed in the experiment were also refined, regarding the materials supplied, the questionnaires, or the duration of the test, among others.

Prior to the activity, the students received a lecture on collaborative systems and awareness, as well as an introductory seminar on the COLLECE system and how it works, with a total duration of 60 min. In this session, the terminology used in the collaborative systems field (CSCW, CSCL, and groupware) was introduced, and the term awareness was defined, as well as some of the mechanisms that support it (telepointers, tele-cursors, session panel, etc.). It was necessary for the participants to be familiar with this terminology, which would be used in the questionnaires to be filled out (pre-test and post-test) to evaluate COLLECE. In any case, images (portions of screenshots of the COLLECE interface) were also included in the questionnaires so that students could identify which element of the interface each question referred to. Furthermore, a document on the use of the COLLECE tool was provided, detailing the steps to follow to connect to the server and to the shared session with their partner, the download of the problem statement to solve, the main components of the interface (areas of the user interface and main buttons), as well as the steps to save the generated solution on the server at the end of the group programming session.

The students, who participated voluntarily in the activity, were scheduled at different times and in groups of two classmates, which were randomly formed by the researchers (as we did not intend to organize for a specific social or educational purpose), to solve a proposed statement jointly using COLLECE. All participants were adequately informed of the purpose of this experiment and provided their informed consent to participate in the activity.

For each student, the test consisted of three *phases* (pre-test, intervention phase, and post-test). Firstly, the participants took a *pre-test* (on paper) by which researchers could gather certain data about the students' profile, including age, prior knowledge, and previous experience (see Section 3.5.1). In this first phase, they also completed a questionnaire, in which they rated their subjective perception of the usefulness (PU), ease of use (PEU), and intention to use (ITU) related to the main UI elements under study.

Then, the participants moved onto the experimental task (*intervention phase*). In this phase, the characteristics of the activity to be developed were explained. The participant pairs were asked to solve a programming problem by applying the *divide and conquer* (D&C) strategy, selected randomly from four different problems available (A-D) (Table 2). To that end, when accessing COLLECE to approach this task, they had to complete a program written in Java, which contained a skeleton of the solution expressed in pseudocode and the parameters to invoke it from the main method. This included a data set in array form that allowed the programmers to test their solution easily. To solve the problem, they had to complete no more than 15–20 lines of code.

As indicated in the previous section, one of the members of the pair occupied the post equipped with the eye tracking device, while the other occupied the one with the video camera. In the case of the former, distance and height adjustments were necessary, as well as calibration of the device. In some couples, there were problems with the calibration of the team member sitting in the eye tracker workstation, so it was necessary to exchange his position with that of his partner until a correct calibration of one of the two team members was achieved.

Once the intervention phase was completed, the participants filled out the *post-test* on paper. In it, they were asked to complete a questionnaire again, this time assessing only the perceived usefulness (PU) and ease of use (PEU) after the group programming activity. Again, these aspects were scored for the nine UI elements of the COLLECE interface. Next, the participants performed a heuristic evaluation of the awareness support provided by the system using the Gutwin and Greenberg's framework. Finally, three open questions were included so that participants could indicate strengths, weaknesses, and suggestions for the improvement of the COLLECE system.

The total duration of the activity in the laboratory was approximately 45 min, which included the completion of the pre-test and post-test questionnaires, the calibration of the eye tracking device, for which a maximum of 10 min was reserved, and the resolution of the pair programming exercise using COLLECE, which had a maximum duration of 25 min.

3.4. Sample

A total of 94 students from the Faculty of Computer Science (ESI) in Ciudad Real, Spain, were recruited to carry out this experiment. The participants were enrolled in the subject *Programming Methodology* of the second year of the Computer Science Engineering Degree and obtained bonus points in the subject for participating in this experiment, whose weight was small in relation to the overall grade for the subject.[6] This experiment was conducted during the second semester of the 2021–2022 academic year.

[6] Providing incentives to recruit experimental subjects is a method commonly used to increase the participation and involvement of participants. In our case, a non-compulsory percentage of active participation in class (10%) is included in the final grade for our degree courses. To obtain this percentage, students participate in various follow-up, assessment, and exercise activities during the course. The use of COLLECE to solve a group programming problem was one of the several activities that students in the course had to carry out to obtain this percentage of the mark. By awarding points for the activity, the researchers aimed to get the students more engaged in the successful completion of the proposed exercise and to get them to take it more seriously. Several authors have studied the ethical aspect of the use of incentives, including monetary ones, and have concluded that their use to recruit and retain subjects is innocuous in most cases (Afkinich and Blachman-Demner, 2020; Grant and Sugarman, 2004; Halpern et al., 2021).

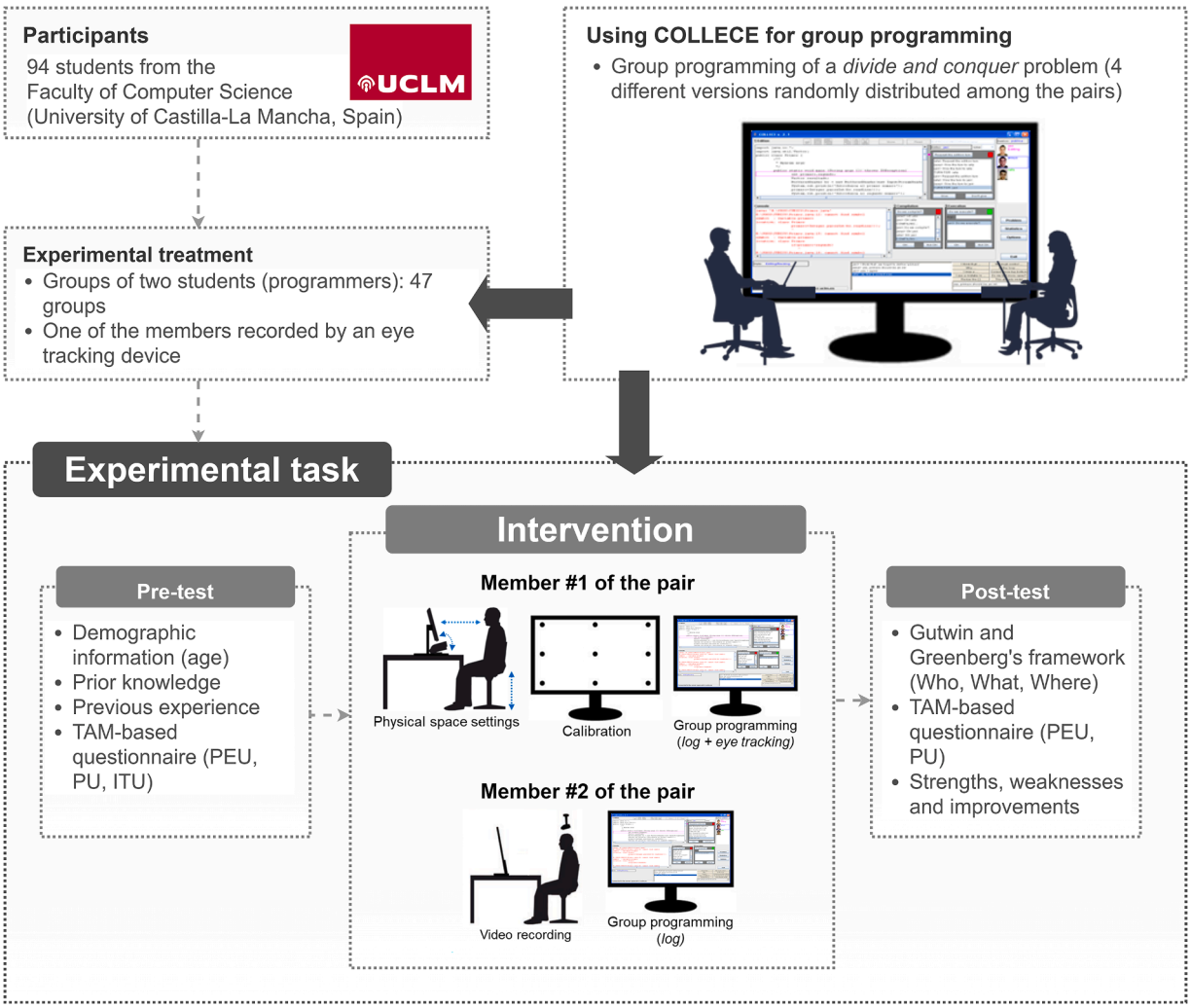

Fig. 3. Overview of the experimental design.

The graph shown in Fig. 4 describes the sample size considered in the different analyses performed to answer the research questions. Although 94 students were recruited and organized in 47 pairs, some of them (a total of 7) did not deliver any of the questionnaires (pre-test or post-test) to be fulfilled, so only the data of 87 participants is considered in the analyses corresponding to RQ1, RQ2, and RQ4.

On the other hand, from the 47 pairs formed, only 33 passed the filter of correct calibration of the eye tracker and obtained accurate recordings. The tool used to record and analyze the data collected by the eye tracker (Tobii Studio[7]) provides an index, in the form of a percentage, of the quality of the recording made. In the case of 14 of the pairs, poor-quality records[8] were obtained, so that these groups could not be considered in the RQ whose response was based on the analysis of the data provided by the device. Therefore, only data recorded of 33 pairs were considered in the RQ1 and RQ2. This represents a loss of almost 30% of the initial sample. Sample loss is a common problem in eye tracking experiments and is estimated to be around 20% (Bojko, 2013). In this case, the loss is higher due to the long duration of the test (approximately 15 min), in which participants may suffer fatigue or distractions, make body or head movements, cover their face or eyes with their hands, as well as adopt postures that may hinder tracking. To reduce the risk of sample loss in eye tracking experiments, it is advisable to over-recruit participants (Bojko, 2013), as was done in this experiment, in which the possibility of losing the records of some of the recruited pairs was already expected. Despite having adopted measures aimed at preventing this problem, such as, for example, warning participants before and during the experiment about the adoption of appropriate postures and the need to avoid sudden movements, this situation is, in many cases, difficult to control.

Finally, there were problems with the automatic logging of the information generated in one of the 47 groups, so only the data from 46 pairs and 85 individuals has been used in the analyses related to RQ4.

3.5. Instruments and measures

This section describes in detail the different instruments and measures used in this experiment. Fig. 5 shows a comprehensive scheme of all the variables collected and calculated. The figure shows, and distinguishes by the corresponding icons, the aspects studied in the experiment, as well as the different types of instruments used to collect the measures (subjective perception questionnaires, metrics provided by the eye tracker, records made by COLLECE's automatic logging module,

[7] https://www.tobii.com/products/software

[8] The software system used to collect eye tracker metrics (Tobii Studio) provides a measure of the quality of the records made. As indicated in the specification of this software (user's manual available in https://connect.tobii.com/s/studio-downloads?language=en_US) and in specific reports dealing with the accuracy and precision of the measurements (Tobii, 2011), it is recommended that this percentage should be above 80%. This was the criterion considered when eliminating records from the final sample.

Table 2
The statements of the four problems posed during the group programming activity.

Problem A: There are two n-integer vectors A and B which satisfy the property that they are distinct component by component up to a given position, and thereafter they are equal component by component. That is, if A and B are distinct up to component 3, that means that A[i] != *B*[i] for *i* = 1,2,3; and that A[i]=*B*[i] for *i* = 4,5,…,n. For example: *A*=[1,3,2,6,5,9] and *B*=[2,7,8,6,5,9]. Write a divide-and-conquer program in Java that calculates the first position in which A and B are equal (in the case of the example, 3).
Problem B: There are two n-integer vectors A and B which satisfy the property that they are equal component by component up to a given position, and thereafter they are different component by component. That is, if A and B are equal up to component 3, that means that A[i]=*B*[i] for *i* = 1,2,3; and that A[i] != *B*[i] for *i* = 4,5,…,n. For example: *A*=[2,3,8,1,4,7,4,9] and *B*=[2,3,8,4,5,6,9,1]. Write a divide-and-conquer program in Java that calculates the first position in which A and B are distinct (in the case of the example, 3).
Problem C: Given a vector of n integers, write a divide-and-conquer program in Java that calculates the maximum element
Problem D: Given a vector of n integers, write a divide-and-conquer program in Java that calculates the minimum element.

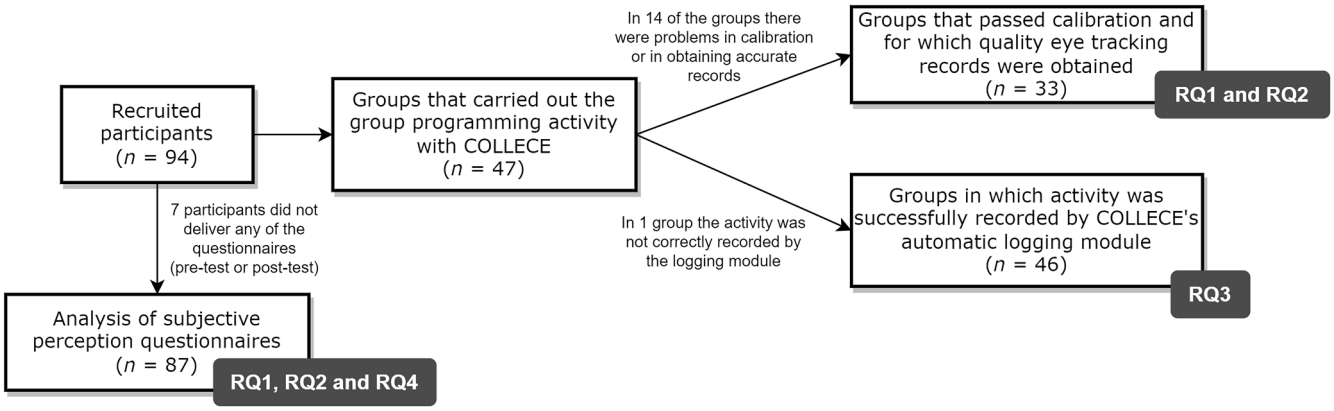

Fig. 4. Different sample sizes considered in the analyses conducted to answer the research questions.

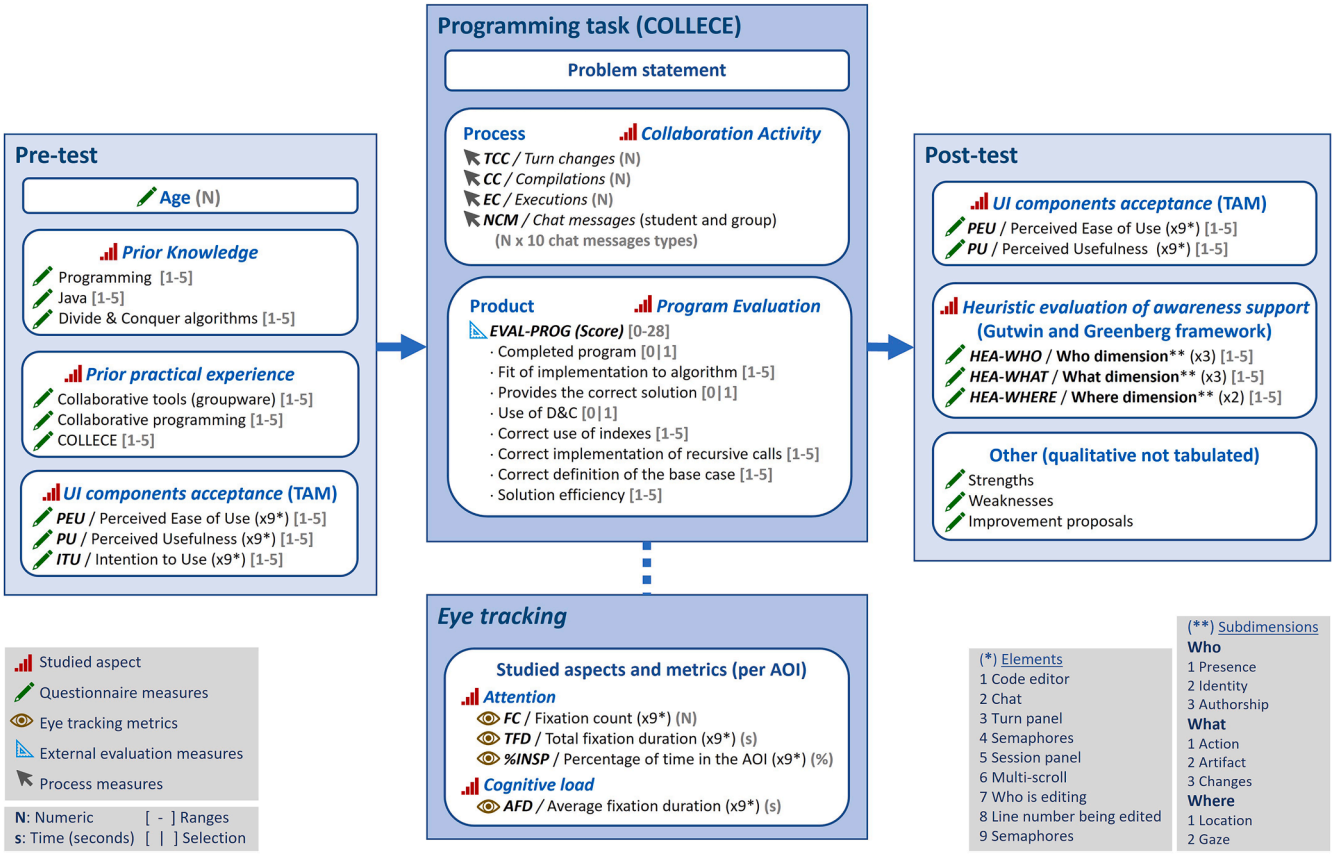

Fig. 5. Overview of the aspects studied and measurements (both subjective and objective) collected, computed or calculated in the experiment.

and external measures or scores given to the generated product). The units and ranges for each of the measures are also shown.

This experimental study combines objective and subjective measures. The following sections describe the instruments used to collect them.

3.5.1. *Subjective measures*

Among the **subjective measures**, the following were collected (see Fig. 5):

- **Participant profile measures** (see *Pre-test* block in Fig. 5). In the phase before intervention (pre-test), it is advisable to collect certain characteristics related to the user's profile. This is done to contrast the influence of personal and demographic factors on test performance. In this case, it has been collected: (a) demographic information (*age*); (b) *prior knowledge* on programming, programming in Java language, and the *divide and conquer* (D&C) programming technique; and (c) *prior practical experience* in the use of collaborative tools, collaborative programming tools, and the use of the COLLECE tool. A five-point response format was used for measuring the participants' prior knowledge and experience.
- **TAM-based questionnaire (TAM)** (see *Pre-test* block in Fig. 5). Participants completed a questionnaire designed to measure their subjective perception regarding the communication, coordination, and awareness mechanisms and widgets included in the COLLECE system (Fig. 1). The questionnaire was based on the TAM (*Technology Acceptance Model*) evaluation framework (Davis, 1993). TAM was the first model to mention psychological factors affecting computer acceptance (Taherdoost, 2018), and research has empirically demonstrated that it is a robust model for understanding end-user adoption of technology and for examining the acceptance of a technological innovation. This framework allows to measure the *perceived usefulness* (PU), the *perceived ease of use* (PEU), and the *intention to use* (ITU) of a system. PU refers to the degree to which a user believes that a technology helps them to achieve their goals and improve their performance. PEU refers to the degree to which a user believes that a technology will be easy to use and learn. Finally, ITU refers to the user's intention to use the technology in the future. The flexibility of TAM to be extended and modified makes it a powerful framework for evaluating software and justifies it being one of the most widely accepted theories among information system researchers for studying the system-acceptance behavior of users (Granić and Marangunić, 2019). In our case it was adapted to measure the PEU, PU and ITU of the communication, coordination and awareness mechanisms and widgets included in the COLLECE system.

  A questionnaire was designed for this purpose, which included a set of items. Each item consisted of a screenshot of the UI element to be evaluated and three questions to be answered on a five-point Likert scale. These questions measured the PU ("*Do you consider the information provided by this graphic element useful during a group programming task?*"), the PEU ("*Do you consider the information provided by this graphic element easy to use and understand?*"), and the ITU ("*Do you think you would use this element during a group programming task?*") for each of the UI elements under study.
- **Awareness support questionnaire** (see *Post-test* block in Fig. 5). In this experiment, a *heuristic evaluation of awareness support* is also carried out. For this purpose, we have chosen to use the Gutwin and Greenberg's framework for heuristic evaluation of collaborative systems (Gutwin and Greenberg, 2002) (see Section 2.2) because of its maturity, widespread use, and simplicity with respect to other proposals (e.g., Antunes et al. 2014, Mantau and Benitti, 2024). From the categories proposed in this framework, the three related to the information a person would require to be aware of the events occurring in a collaborative workspace in real time, i.e., the elements of *workspace awareness* related to the present (*Who, What,* and *Where*), were considered (Tam and Greenberg, 2006). Table 3 shows the questions proposed by these authors to measure each of the subdimensions or informational elements contained in each of these categories, which had to be answered by the subjects in a five-point response format (ranging from 1, "very poor support", to 5, "excellent support") (HEA-WHO, HEA-WHAT, HEA-WHERE).

3.5.2. Objective measures

Among the **objective measures** collected and calculated, we can mention the following (see Fig. 5):

Table 3
Dimensions, subdimensions, and questions proposed by Gutwin and Greenberg's framework (Tam and Greenberg, 2006).

Dimension	Subdimension	Questions
Who	Presence	*Is anyone in the workspace?*
	Identity	*Who is participating in the group programming activity? Who are they?*
	Authorship	*Who edited which part of the code?*
What	Action	*What is the peer doing? What are the activities they are currently doing?*
	Artifact	*What part of the code is the peer working on?*
	Changes	*What changes is the peer making to the code?*
Where	Location	*Where is the peer editing?*
	Gaze	*Where is the peer looking? Where in the code is his or her focus of attention?*

Table 4
Correction rubric, including characteristics of the program solution to be scored and range of points to be assigned to each of them. The product measure is obtained as the sum of the scores given to each of the aspects considered, with a maximum mark of 28.

Rubric	
Aspect to evaluate	**Score range (points)**
Completed program	0/1
Fit of implementation to pseudocode algorithm	up to 5
Providing the correct solution	0/1
Use of D&C technique	0/1
Correct use of indexes	up to 5
Correct implementation of recursive calls	up to 5
Correct definition of the base case	up to 5
Solution efficiency	up to 5
EVAL-PROG (Program evaluation)	**0 to 28**

- **Product measure** (see *Programming task* block in Fig. 5). As a result of the collaborative activity, a source code (Java program) will be obtained as a final product, which will be evaluated and from which a *score* will be obtained (EVAL-PROG). This rating is a measure of team performance and has a value between 0 and 28 points, obtained from the ratings given by an evaluator to a series of characteristics of the program (Table 4)[9]: *completed program, fit of implementation to pseudocode algorithm, providing the correct solution, use of D&C technique, correct use of indexes, correct implementation of recursive calls, correct definition of the base case*, and *solution efficiency*.

- **Process measures** (see *Programming task* block in Fig. 5). COLLECE incorporates a *logging* module to record the use of the decision-making and polling tools included in the system user interface as an interaction count of turn changes (TCC), compilations (CC), and executions (EC) (see Bravo et al. 2008, to explore further this analysis subsystem). It also allows recording the *messages exchanged in the chat* (NCM) in each programming session in group, and those produced by each of its members. In both cases, the content of the generated message is recorded, as well as its type (*cmt: chat message type*), with 10 possible types of messages corresponding to the sentence openers available in the COLLECE's structured chat: "*I think that …*", "*Why …*", "*I miss a …*", "*I see a mistake in …*", "*Revise the {}*", "*We must control …*", "*In the loop …*", "*Correct errors top-bottom.*", "*Do we test more cases?*" and "*This already works.*".
- **Eye tracking metrics** (see *Eye tracking* block in Fig. 5). Finally, thanks to the use of the eye tracker device, a series of metrics are

[9] The characteristics to be evaluated in the product obtained in the programming task (and included in the correction rubric) were proposed by the professors of the "Programming Methodology" subject, in which the experiment was carried out, and are based on their experience and criteria when evaluating their students' deliverables.

Table 5
Eye tracking measures (definition and interpretation) (Bojko, 2013; Poole and Ball, 2006).

Measured aspect	Eye tracking metric	Definition (metric calculation)	Metric interpretation
Attention	*Fixation Count* (FC) - *count*	This metric measures the number of times the participant fixates (number of fixations) on an AOI or an AOI group. If during the recording the participant leaves and returns to the same media element, then the new fixations on the media will be included in the calculations of the metric.	This metric helps measure how much attention a viewer dedicates to specific elements within a visual stimulus in a user interface. Higher fixation counts within an AOI may indicate that the element was more noticeable, useful, or important.
	Total Fixation Duration (TFD) - *seconds*	The Total Fixation Duration represents the cumulative time (in seconds) that the eyes spend fixating on a particular area or across the entire visual scene. This metric measures the sum of the duration for all fixations within an AOI (or within all AOIs belonging to an AOI group).	Time spent inspecting a specific area of the user interface or widget. It provides insights into how long a viewer concentrates on specific AOI. Longer fixation durations within an AOI may indicate that the element was more salient, engaging or useful.
	Percentage of inspection time (%INSP) *in an AOI - percentage*	Total Fixation Duration on an AOI (TFD) divided by the total time of inspection of the overall user interface.	The ratio of total time spent on task-relevant AOIs relative to time spent on unspecified areas or task-irrelevant AOIs. This metric gives a measure of the usefulness of those to which a higher percentage of inspection time has been devoted.
Cognitive load	*Average Fixation Duration* (AFD) - *seconds*	The Average (or Mean) Fixation Duration is the average length of time that the eyes fixate on a single point during a given observation. It is calculated by dividing the Total Fixation Duration by the Fixation Count.	A longer duration indicates that the participant needs more time to understand the individual objects. This may indicate difficulty in extracting information, i.e., the need for more cognitive effort to process it. Conversely, shorter average fixation durations could indicate quicker or easier information processing, suggesting lower cognitive load.

collected to measure the *attention* and *cognitive load* imposed by different components and widgets available in the COLLECE interface. For that purpose, one of the first steps to be taken is the definition of the so-called *areas of interest* (AOIs) of the stimulus (in our case, the user interface) under evaluation (see Section 3.5.3). The AOIs are those that are relevant to the evaluator in the context of certain evaluation objectives or research questions. For each of these AOIs, a more detailed analysis will be made by calculating some metrics (Sharafi et al., 2015a). From the set of metrics that can be obtained from an eye tracker session, those based on the so-called fixations stand out. A *fixation* refers to a period during which the eye is relatively stable and focused on a particular point in the visual field. From fixations, metrics can be calculated to infer interest or attention, efficiency in locating and recognizing a given element, or mental effort or cognitive load imposed by the content of a particular AOI, among others (Bojko, 2013; Poole and Ball, 2006).

Table 5 compiles the metrics considered in this experiment, focused on measuring *attention* and *cognitive load* (Sharafi et al., 2020), and that can be used to respond to the raised RQs. *Attention* metrics commonly include the number of fixations on an AOI (FC) generated in an AOI, the total time spent inspecting an AOI or widget (TFD), and the percentage of total time spent inspecting one AOI versus the rest (%INSP). All these metrics are related to interest, attention, and indirectly usefulness, as we show more interest or visually attend more to areas of the screen that are more helpful in the performance of a given task. On the other hand, there is another set of metrics that allow for measuring the user's performance in carrying out a task. Some of them indicate the mental effort and *cognitive* processing associated with the visual analysis of an image or interface. This is the case of the average fixation duration (AFD) in an AOI. Longer fixation durations indicate that more time is required to understand individual objects, i.e., there is greater difficulty in extracting information.

3.5.3. Definition of the areas of interest

In eye tracking sessions, one of the main steps is the definition of the so-called *areas of interest* (AOI) of the image or user interface under evaluation. These areas of interest are for the researcher, and they are the ones that will be analyzed more exhaustively through the calculation of certain metrics (Molina et al., 2024). In the case of the COLLECE evaluation, the AOIs delimit the area occupied by the widgets or interface components under study (Fig. 6).

Table 6 describes the nine COLLECE interface elements under study (corresponding to the defined AOIs, drawn in Fig. 6). For each of them, subjective (PEU, PU, and ITU) and objective (eye tracking metrics) measures were obtained (see Fig. 5). The table also includes the classification of these AOIs into collaboration support (shared context, communication tool, collaboration tool, or awareness support mechanism) and the support they provide to the dimensions of Gutwin and Greenberg's framework.

3.5.4. Measurement model to answer the research questions

Table 7 summarises the aspects studied to address the different research questions, and the measures and metrics associated with them, together with the instruments used to get or calculate each measure or metric. The *questionnaire measures* were collected in paper format at the pre-test and post-test stages (see Fig. 5); the *eye tracking metrics* are calculated by the Tobii software (Tobii Pro, 2016); the *process measures* are obtained from the COLLECE's logging module; and the *product measures* are assigned by an evaluator according to a rubric (Table 4). Fig. 5 shows the ranges and data types of all these variables.

3.6. Data collection and preprocessing

As indicated previously, the Tobii Studio software version 3.4.8 was used to calibrate the device, record the visual behaviour of the participants, define the AOIs, and then calculate the metrics for each of them (Tobii Pro, 2016). To define the final set of records to be analyzed, the ones of participants where calibration had not been performed accurately or where the recorded data were incomplete or of poor quality were eliminated (see Section 3.4).

It is also necessary to comment that the visual stimulus used in this experiment, consisting of a desktop graphical user interface whose use is dynamic and different for each participant, made it difficult to analyze and collect the metrics provided by the eye tracking device. When recording the eye activity of a user interacting with a user interface (application or web page), it must be kept in mind that the user's interactions are free and different in each case, resulting in unpredictable what may appear on the screen at each instant of time. Thus, for example, because of the interaction with a menu or dialog box, regions of the screen containing AOIs may be hidden. Therefore, in these cases, individualized analysis of the records made for each of the participants

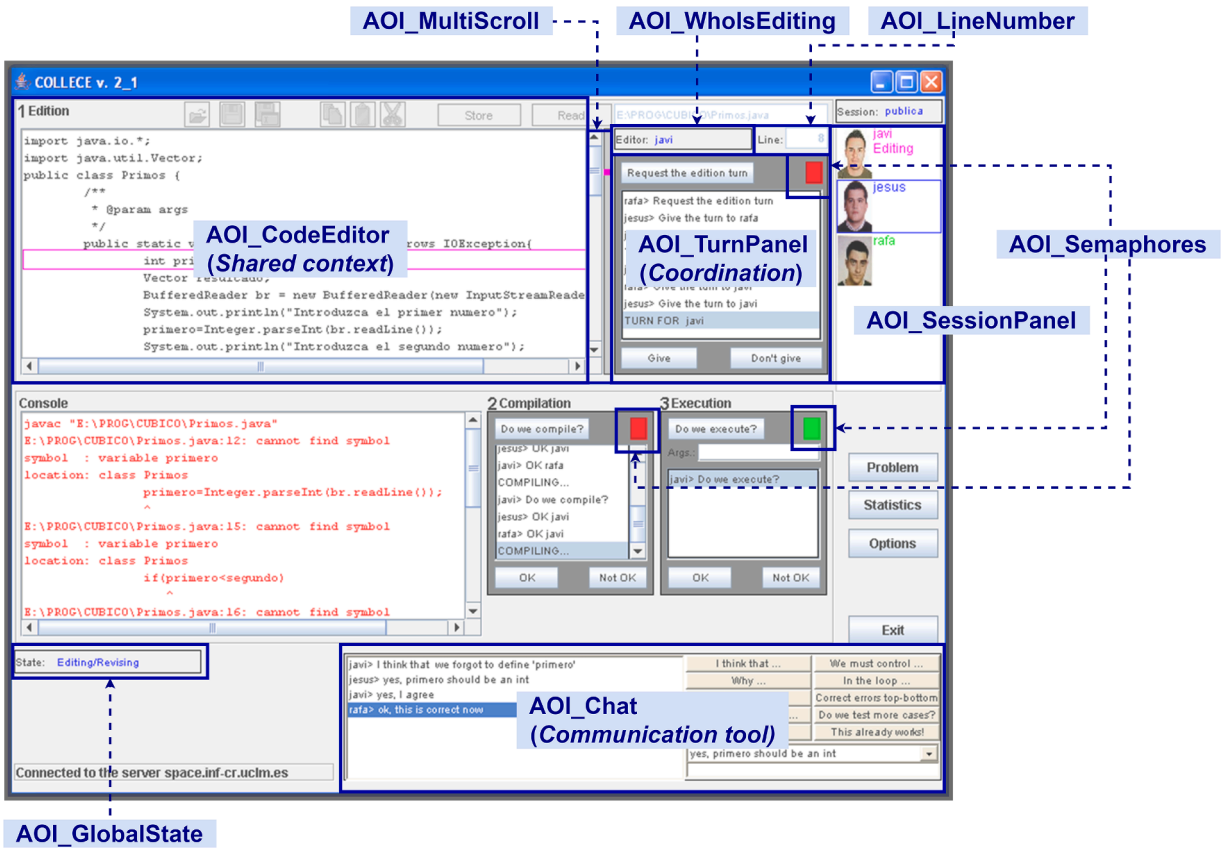

Fig. 6. Definition of the AOIs in the COLLECE interface, which delimit the main components supporting communication, coordination and awareness.

in the experiment is required, segmenting the record into portions in which the interface does not suffer variations and collecting for each segment the metrics of interest for the AOIs displayed on the screen during that subperiod of time (Molina et al., 2024). Once this data extraction is done at the frameset level, the data can be reassembled to obtain metrics corresponding to the complete development of the activity. This process of filtering, segmenting, and metric compilation complicates the process considerably and demands greater resources (time and effort) for processing and extracting information of interest. All this individualized treatment of the records and preprocessing of the information was carried out to obtain the final values of the metrics recorded by the eye tracker.

On the other hand, it should be clarified that, although the entire COLLECE interface is a dynamic stimulus, the AOIs it contains are not. Thus, for example, although the content to be displayed in the code editor is dynamic (and depends on the movement of the scroll bar), since in the study carried out we were only interested in the attention paid to that area as a whole and not to the specific content contained therein, it was not necessary to re-segment the records to perform a low-granularity analysis in this case. Regarding the AOI containing the three semaphores included in the interface, it was necessary to aggregate, i.e., add the values of the metrics obtained for each of them, to obtain the final measures of this group of visual elements.

The calculated metrics provided by the Tobii Studio software were compiled in a spreadsheet, which was completed with the information recorded by the automatic logging module, including counts of the types of contributions made in the chat by everyone, as well as the interactions with the compilation and execution components. Finally, the sheet continued to be completed with the responses gathered in the pre- and post-tests (which had been collected on paper) and the score of the exercise carried out.

3.7. Data analysis

This section presents the analysis of the data gathered in the study.[10] The total number of participants recruited was 94 students, but as described in Section 3.4, different sample sizes are used in the analyses described in this section.

Participants were aged between 19 and 33 years, with an average age of 21.8 (sd=3.3). As for their profile (Table 8), the students considered that they had average knowledge in programming and Java (md=3.00), although somewhat lower in the *divide and conquer* technique (md=2.00), which had been taught in the course in the previous weeks. As for their practical experience, both in the use of collaborative tools in general and group programming, as well as in the specific use of COLLECE, this was scored around 2.00.

Participants took, on average, 18.43 min (sd=4.37) to complete the programming task. Next, a description of the analyses carried out to seek to answer the research questions posed is presented.

RQ1. ***What are the most useful collaboration support mechanisms (communication, coordination, and awareness) for programmers in the context of distributed collaborative programming tasks in COLLECE?***

[10] As indicated, the questionnaires (*pre-test* and *post-test*) include items with a five-point Likert response format. The presentation of the results in this section follows the recommendations given in the literature (Alkharusi, 2022; Harpe, 2015) concerning the analysis and interpretation of a five-point agreement response scale, according to which "*since the response format contained at least five categories, it is possible to treat them as continuous variables and report means and standard deviations for each item.*".

Table 6
Description of the areas of interest (AOI) defined in the COLLECE interface and their support for the dimensions proposed by Gutwin and Greenberg.

AOI identification	Elements in the COLLECE user interface	Collaboration support (tools and widgets)	Dimensions of Gutwin & Greenberg's framework supported
AOI_CodeEditor	Code editing area. It constitutes the shared context of the group activity. It includes visual indicators of the line on which the peer is editing, identified by his/her color (tele-cursors).	Shared context	Who? What? Where?
AOI_Chat	Area of the interface that delimits the synchronous communication tool or chat. It incorporates the possibility of using sentence openers to specify the type of contributions in the chat.	Communication	Who? What?
AOI_TurnPanel	Area of the interface that delimits the coordination support tool. It allows users to request and give up the editing turn (floor control) of the shared context.	Coordination	Who? What?
AOI_Semaphores	Semaphore-type visual indicators (red/green codes) to control turns in the decision-making processes. This AOI delimits the three interface areas occupied by the three semaphores included in the COLLECE user interface (Fig. 1).	Coordination / Awareness	Who? What?
AOI_SessionPanel	Area of the interface that displays the set of users who are participating in the programming activity, identified by means of a specific color.	Awareness	Who? What?
AOI_MultiScroll	Area of the user interface occupied by the scroll bar, whose appearance is enriched with visual clues (color coding that uniquely identifies each member of the group) and which indicates in which relative position in the shared space peers are working.	Awareness	Who? What? Where?
AOI_WhoIsEditing	Interface area that provides information about who is editing.	Awareness	Who? What?
AOI_LineNumber	Area of the interface showing the number of the code line being edited.	Awareness	What? Where?
AOI_GlobalState	Interface area displaying the current group activity being carried out in the collaborative system.	Awareness	What?

Table 7
Measures and metrics used to answer the research questions.

Research question	Instrument	Studied aspect	Measure/Metric
RQ1	Pre-test questionnaire (TAM)	User perception of UI components	Perceived usefulness (PU) Intention to use (ITU)
	Eye Tracking	Attention	Fixation count (FC) Total Fixation Duration (TFD) Percentage of inspection time (%INSP) in an AOI
	Pre-test questionnaire	Participants' profile	Prior knowledge Prior practical experience
RQ2	Post-test questionnaire (TAM)	User perception of UI components	Perceived ease of use (PEU)
	Eye Tracking	Cognitive load	Average fixation duration (AFD)
	Pre-test questionnaire	Participants' profile	Prior knowledge Prior practical experience
RQ3	Process indicators (automatic logging module)	Collaborative activity (communication and coordination)	Coordination activity on turn changes (TCC) Coordination activity on program compilations (CC) Coordination activity on program executions (EC) Communication activity (NCM)
	Product evaluation	Solution/Program quality	Score representing the evaluation of the solution program (EVAL-PROG)
RQ4	Post-test questionnaire	Heuristic evaluation of awareness support	Heuristic evaluation of the awareness support according to the Who dimension (HEA-WHO) Heuristic evaluation of the awareness support according to the What dimension (HEA-WHAT) Heuristic evaluation of the awareness support according to the Where dimension (HEA-WHERE)
	Product evaluation	Solution/Program quality	Score representing the evaluation of the solution program (EVAL-PROG)

To answer the first research question (**RQ1**), we focus on three aspects (*user perception* of UI components, *attention*, and *participant's profile*) and seven measures (see Fig. 5 and Table 7).

To determine which elements are perceived as most ***useful*** by the participants in this experiment, the answers given in the questionnaires based on the TAM framework and, specifically, in the variables that measure the *perceived usefulness* (PU) and the *intention to use* (ITU) of the different elements included in the COLLECE interface (see Fig. 1) were analyzed. As can be seen in Table 9, before performing the experimental task, the elements best rated in terms of their usefulness (PU) were the session panel (m = 4.31, sd=0.80), the code editor in which the line being edited by the peer was highlighted in its identifying color (m = 4.22, sd=0.72), followed by who is editing (m = 4.16, sd=0.89) and which line (m = 4.08, sd=0.89). And, consistently, these four UI elements were also the ones that scored best on the variable measuring the *intention to use* (ITU), although in this case the highest rated element was the widget that shows who is currently editing the code (m = 4.18, sd=0.91), followed by the code editor (m = 4.10, sd=0.82).

Once the experimental task was completed, the participants again rated the PU of the main components of the UI. In this case, the elements that obtained the highest scores were, once again, the editing area (m = 4.61, sd=0.69), with an increase that is considered significant, and the traffic lights, which allowed coordination of group activity (m = 4.10, sd=0.94) (Fig. 7a). Most of the awareness items decreased in score between pre-test and post-test and did so significantly (as can be seen in the last column of Table 9) except the multi-scroll bar, which remained at similar values.

By recording the participants' eye movements, it is possible to determine which elements of the UI were most ***attended*** *to*. According to several authors (Bojko, 2013; Sharafi et al., 2020), some of the most commonly used metrics for measuring attention are the fixation count (FC), the total time spent inspecting a given AOI, which can be measured as the sum of the duration of all fixations on that AOI (TFD), and the percentage of time spent attending to that AOI in relation to the total

Table 8
Descriptive statistics of the profile features of participants in the empirical study. Values for $n = 87$ participants.

Prior knowledge and Practical experience	Mean [a] *m (sd)*	Median *md*
Prior knowledge in programming	3.22 (0.71)	3.00
Prior knowledge in Java	3.41 (0.69)	3.00
Prior knowledge in *divide and conquer*	2.30 (0.72)	2.00
Practical experience in the use of collaborative tools	1.98 (0.91)	2.00
Practical experience in the use of collaborative programming tools	1.69 (0.72)	2.00
Practical experience in the use of COLLECE	1.78 (0.71)	2.00

[a] We show the mean and, in parentheses, the standard deviation.

time or inspection time (%INSP) (see Fig. 5 and Table 5). Table 10 shows the average values of these metrics for the AOIs defined in the COLLECE interface. As can be seen, the elements that obtained the highest values in all these metrics were the code editor, the synchronous communication tool (chat), and the turn assignment panel (Fig. 7b). Among the awareness support elements, the most attended were the session panel and the area indicating who is editing the code at any given moment.

Additionally, the information related to the participants' profile was **correlated** with the subjective measures obtained, and it was found that there was a statistically significant correlation ($p < 0.05$) between the perception of having greater prior knowledge of programming and the usefulness of the code editor, although the relationship is weak ($r = 0.243$). The correlation between a higher intention to use this specific component of the interface and having more experience with collaborative systems was statistically significant but weak relationship ($r = 0.255$, $p < 0.05$).

When **correlating** the *participant's profile* and the *attention* given to the different areas of the UI (measured by the percentage of time dedicated to its visual inspection), it was found that those who scored higher in previous Java knowledge dedicated less time to inspecting the chat ($r=-0.375$, $p < 0.01$), as well as the session panel ($r=-0.293$, $p < 0.05$). On the other hand, those who indicated having more experience in the use of collaborative tools consulted the semaphores less ($r=-0.312$, $p < 0.05$), and those who had more experience in the use of group programming tools, attended less to the line that was being edited at each moment ($r=-0.311$, $p < 0.05$).

RQ2. ***What are the easiest collaboration support mechanisms to use in the context of distributed collaborative programming tasks in COLLECE?***

To answer the second research question (**RQ2**), we focus on three aspects (*user perception* of UI components, *cognitive load*, and *participant's profile*) and four measures (see Fig. 5 and Table 7).

To determine which elements were perceived as the *easiest to use* by the participants in this experiment, the PEU of the TAM framework for the different elements of the COLLECE interface was analyzed. As can be seen in Table 11, before performing the experimental task, the highest rated elements were the session panel ($m = 4.38$, sd=0.78), the interface element indicating the line being edited ($m = 4.30$, sd=0.72), the indicator of who is doing it ($m = 4.28$, sd=0.83), and the code editor ($m = 4.25$, sd=0.75). On the other hand, the element that a priori they valued as the most complex to use was the panel for requesting and giving up editing turns ($m = 3.47$, sd=0.99).

The measures obtained in the post-test with respect to this aspect, once COLLECE was used in the collaborative programming task, underwent some variations. Again, the element that obtained the highest rating was the session panel ($m = 4.20$, sd=0.83), followed by the code editor ($m = 4.16$, sd=0.64) and the traffic light-based awareness mechanism ($m = 4.16$, sd=0.91) (Fig. 8a). On the other hand, the evaluation of the floor control panel increased significantly, no longer being the most complex element to use.

Regarding the *cognitive load* imposed by the different components of the COLLECE user interface, measured by the eye tracker metric average fixation duration (AFD), it was found that the elements that obtained the lowest values were the semaphores and the line number that is currently being edited (Table 12). A quick visual scan of these elements of the UI provided the information the user needed to obtain. On the other hand, the elements that imposed the highest cognitive load were the chat and the code editor, which makes sense since users must look at and try to understand the content displayed in both interface components (Fig. 8b).

With respect to the analysis of the correlations between the participant profile variables and the responses given in the TAM-based questionnaire items, it was found that subjects who scored high in perceived prior knowledge of programming also considered the AOI of the code editor as an easy-to-use item ($r = 0.260$, $p < 0.05$). On the other hand, those who reported higher prior experience in using collaborative systems considered the code editor ($r = 0.229$, $p < 0.05$) and semaphores (r

Table 9
Descriptive statistics of the PU and ITU measures of COLLECE user interface elements and comparative analysis of the PU variable (pre-test and post-test). Values for $n = 87$ participants.

Area of Interest (AOI)	PU [a] *pre-test m (sd)*	ITU [a] *pre-test m (sd)*	PU [a] *post-test m (sd)*	Pre-post comparative analysis for the PU variable			
				Corr. PU [b]	*p-value*	Mean difference [c] *p-value*	Effect size [d] *r*
AOI_CodeEditor	4.22 (0.72)	4.10 (0.82)	4.61 (0.69) ↑	0.389**	<0.001	<0.001	0.45
AOI_Chat	3.90 (1.01)	3.92 (1.11)	3.93 (1.33) ↔	0.380**	<0.001	0.601	0.06
AOI_TurnPanel	3.62 (0.93)	3.40 (1.05)	3.56 (1.15) ↔	0.490**	<0.001	0.645	0.05
AOI_Semaphores	4.07 (0.89)	4.06 (0.99)	4.10 (0.94) ↔	0.414**	<0.001	0.787	0.03
AOI_SessionPanel	4.31 (0.80)	4.09 (0.95)	3.84(1.04) ↓	0.553**	<0.001	<0.001	0.48
AOI_MultiScroll	3.87 (0.90)	3.74 (1.02)	3.87 (1.26) ↔	0.484**	<0.001	0.959	0.01
AOI_WhoIsEditing	4.16 (0.89)	4.18 (0.91)	3.82 (1.05) ↓	0.216*	0.044	0.010	0.28
AOI_LineNumber	4.08 (0.89)	4.08 (0.88)	3,44 (1.37) ↓	0.583**	<0.001	<0.001	0.49
AOI_GlobalState	3.70 (0.89)	3.56 (1.08)	3.30 (1.09) ↓	0.314**	0.003	0.001	0.34

[a] Mean and, in parentheses, the standard deviation is shown. In the post-test results it is indicated whether the value of the statistic has increased (↑), decreased (↓) or remained the same (↔).
[b] Calculated statistic: Spearman's Rank Correlation Coefficients. The level of significance (alpha value):
** The correlation is significant at the 0.01 level (bilateral).
* The correlation is significant at the 0.05 level (bilateral).
[c] Calculated statistic: Paired samples Wilcoxon-test (also known as Wilcoxon signed-rank test). The level of significance (alpha value) is 0.05.
[d] To quantify the effect size, reference is taken from (Cohen, 1992), which indicates: no effect or very small (*<0.1);* small effect (0.10- <0.3), medium or moderate effect (0.30-<0.50) and large effect (≥0.50).

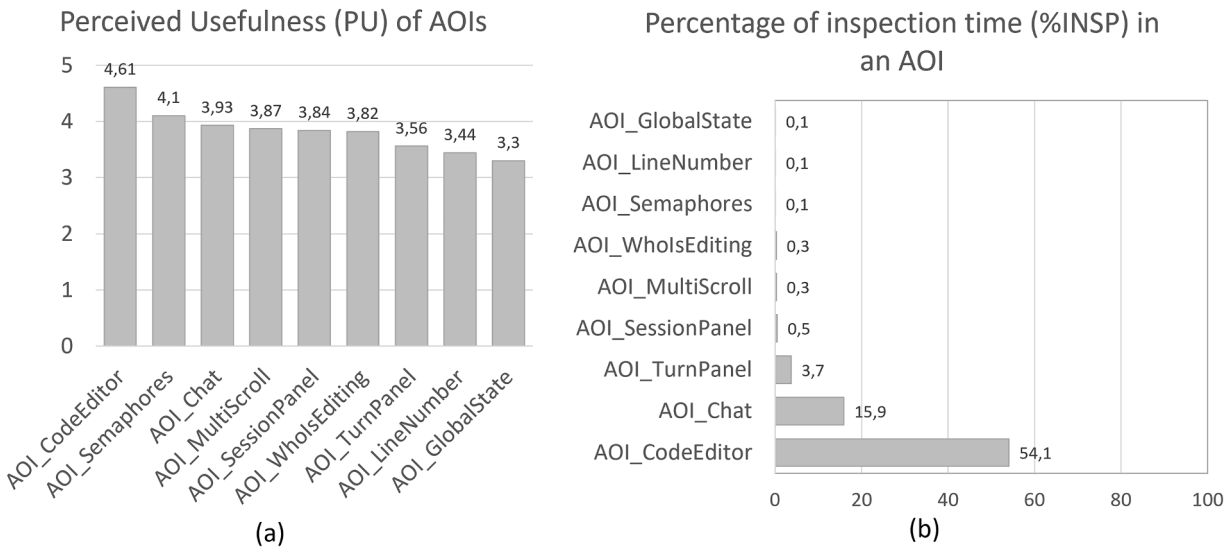

Fig. 7. **(a)** Ranking of AOIs from highest to lowest perceived usefulness (PU) by participants after performing the collaborative programming activity with COLLECE. Results of the data analysis of all participants ($n = 86$). **(b)** Ranking of AOIs according to participants' visual attention (eye tracker metric "percentage of inspection time in an AOI"). Result of the analysis of the data of the participants whose activity was recorded by the eye tracker device ($n = 33$).

Table 10
Attention metrics values provided by the eye tracker. Values for $n = 33$ participants, whose behavior was recorded by the eye tracker and for which accurate records were obtained.

Area of Interest (AOI)	Fixation count (FC) - *count*	Total Fixation Duration (TFD) - *seconds*	Percentage of inspection time in an AOI (%INSP) - *percentage*
AOI_CodeEditor	1511.1	405.1	54.1
AOI_Chat	430.1	129.9	15.9
AOI_TurnPanel	148.6	34.2	3.7
AOI_Semaphores	3.0	0.6	0.1
AOI_SessionPanel	22.2	5.1	0.5
AOI_MultiScroll	10.6	3.2	0.3
AOI_WhoIsEditing	12.4	2.7	0.3
AOI_LineNumber	2.8	0.6	0.1
AOI_GlobalState	3.1	0.7	0.1

$= 0.231, p < 0.05$) as easy-to-use elements, while those with higher prior experience in collaborative programming systems also found the turn requests and assignment system to be an easy component ($r = 0.239, p < 0.05$). In any case, although statistically significant correlations have been obtained, in most cases these correlations are not very strong.

Correlating the profile data with indicators of cognitive effort (measured using the AFD eye tracker metric), it was found that there was a negative correlation between prior Java knowledge and the cognitive load of the AOI_LineNumber ($r = -0.303, p < 0.05$).

RQ3. *Is there a relationship between using certain components of the COLLECE user interface and obtaining a better product?*

To answer the third question (**RQ3**), we study first the *collaboration activity* and then the *program evaluation*. The data collected through the automatic logging module incorporated in the COLLECE tool is used, considering in this case the data recorded for the 46 pairs that carried out the activity as well as the 85 participants.

The COLLECE's log module collects information on the use of the

Table 11
Descriptive statistics of the PEU measure of COLLECE user interface elements and comparative analysis of the PEU variable (pre-test and post-test). Values for $n = 87$ participants.

Area of Interest (AOI)	PEU [a] *pre-test* *m (sd)*	PEU [a] *post-test* *m (sd)*	Pre-post comparative analysis for the PEU variable			
			Corr. PEU [b]	*p-value*	Mean difference *p-value* [c]	Effect size [d] *r*
AOI_CodeEditor	4.25 (0.75)	4.16 (0.64) ↓	0.360**	<0.001	<0.001	0.41
AOI_Chat	3.89 (1.08)	3.57 (1.42) ↓	0.235*	0.036	0.092	0.18
AOI_TurnPanel	3.47 (0.99)	3.75 (0.99) ↑	0.395**	<0.001	0.023	0.24
AOI_Semaphores	4.16 (0.79)	4.16 (0.91) ↔	0.294**	0.006	0.812	0.09
AOI_SessionPanel	4.38 (0.78)	4.20 (0.83) ↓	0.355**	<0.001	0.059	0.20
AOI_MultiScroll	4.01 (0.84)	4.13 (1.04) ↑	0.598**	<0.001	0.154	0.15
AOI_WhoIsEditing	4.28 (0.83)	4.05 (0.85) ↓	0.367**	<0.001	0.026	0.24
AOI_LineNumber	4.30 (0.72)	3.98 (1.09) ↓	0.510**	<0.001	0.004	0.31
AOI_GlobalState	3.84 (0.93)	3.71 (1.03) ↓	0.519**	<0.001	0.277	0.12

[a] Mean and, in parentheses, the standard deviation are shown. In the post-test results it is indicated whether the value of the statistic has increased (↑), decreased (↓) or remained the same (↔).
[b] Calculated statistic: Spearman's Rank Correlation Coefficients. The level of significance (alpha value):
** . The correlation is significant at the 0.01 level (bilateral).
* . The correlation is significant at the 0.05 level (bilateral).
[c] Calculated statistic: Paired samples Wilcoxon-test (also known as Wilcoxon signed-rank test). The level of significance (alpha value) is 0.05.
[d] To quantify the effect size, reference is taken from (Cohen, 1992), which indicates: no effect or very small (<0.1); small effect (0.10- <0.3), medium or moderate effect (0.30-<0.50) and large effect (≥0.50).

coordination and communication support tools, recording the number of interactions with the different voting and group decision-making tools as well as the count of messages generated in the chat at the user and group level. In the latter case, the number of messages per group and user is also collected for the subtypes supported by means of the sentence openers available in the structured chat (see Section 2.4). Table 13 shows the average values of the count of interactions and messages generated. As can be seen in the results obtained, the element with which users most interact is the tool for the exchange of editing turns between the members of the pair ($m = 4.33$, sd=2.17). Regarding the type of messages most used, the subtype *cmt1* (sentence starter "*I think that…*"), which is the most generic type of sentence opener, stands out.

The product resulting from the collaborative programming activity, evaluated by an expert using the rubric shown in Table 4 as stated before, obtained an average score of 15.23 points.

The correlational analysis between the score obtained (EVAL-PROG), at group level and per participant, and the use of communication and coordination tools shows that the groups that compiled and executed the program more times obtained a higher score ($p < 0.01$) (Table 14). On the other hand, those who communicated more also made a greater number of exchanges in the editing turn ($p < 0.05$).

Table 15 shows the correlations between these product quality criteria and the interaction count performed using the coordination and communication tools. Furthermore, regarding the latter aspect, they have also been correlated with the type of specific contribution made in the chat (using sentence openers). As can be seen in Table 15, the participants who obtained higher scores on all the solution quality indicators also made more intensive use of the compilation and execution tools. On the other hand, those who did not follow the correct approach (application of the *divide and conquer* technique) in solving the problem posed emitted more messages of the type "*In the loop…*", a fact that makes sense since this approach does not make use of loops but of recursive calls to solve the problem posed. Those who made more use of the "*Why…*" statement opener also scored better in the fit of the implementation to the algorithm, as well as in the use of the *divide and conquer* technique and the definition of the base case. On the other hand, those who used the sentence starter "*I see a mistake in…*" also scored better in solution efficiency and in the fit of the implementation to the algorithm.

To determine whether there was any correlation between the quality of the product generated and the *perceived usefulness* and *attention* paid to the different components of the interface, a correlation analysis was performed between the variables that allowed assessing these aspects. Thus, it was found that there was a positive, but weak, correlation between having rated the code editor as more useful and the score obtained ($r = 0.22$, $p < 0.05$) and most of its quality indicators. On the other hand, when correlating the score obtained with the measures of attention to the different components of the user interface (percentage of inspection time and fixation count), a negative correlation was obtained between the score (and most of the solution quality sub-indicators) and the visual attention paid to the session panel ($p < 0.01$).

When reviewing the correlations between the user profile variables, the score obtained, and the use of coordination and communication tools, some correlations were found. Those participants who scored higher on prior programming knowledge performed more compilations ($r = 0.307$, $p < 0.05$) and executions ($r = 0.316$, $p < 0.05$). A correlation was also found between having indicated a higher knowledge of Java and the number of compilations performed ($r = 0.320$, $p < 0.05$. On the other hand, students with a greater knowledge of the *divide and conquer* technique obtained, as expected, a higher score in the activity ($r = 0.301$, $p < 0.05$), finished the program ($r = 0.348$, $p < 0.05$) and applied this programming technique correctly ($r = 0.478$, $p < 0.01$). In addition, the solution generated was more efficient ($r = 0.324$, $p < 0.05$), with a better code fit to the algorithm ($r = 0.3.08$, $p < 0.05$) and with the correct use of recursive calls ($r = 0.332$, $p < 0.05$). On the other hand, those who had a higher level in Java scored better in the criterion related to the correct use of indexes ($r = 0.302$, $p < 0.05$). It should be noted

Table 12
Cognitive load metric values provided by the eye tracker. Values for $n = 33$ participants, whose behavior was recorded by the eye tracker.

Area of Interest (AOI)	Average fixation duration (AFD) - *seconds*
AOI_CodeEditor	0.27
AOI_Chat	0.29
AOI_TurnPanel	0.22
AOI_Semaphores	0.14
AOI_SessionPanel	0.20
AOI_MultiScroll	0.26
AOI_WhoIsEditing	0.21
AOI_LineNumber	0.15
AOI_GlobalState	0.20

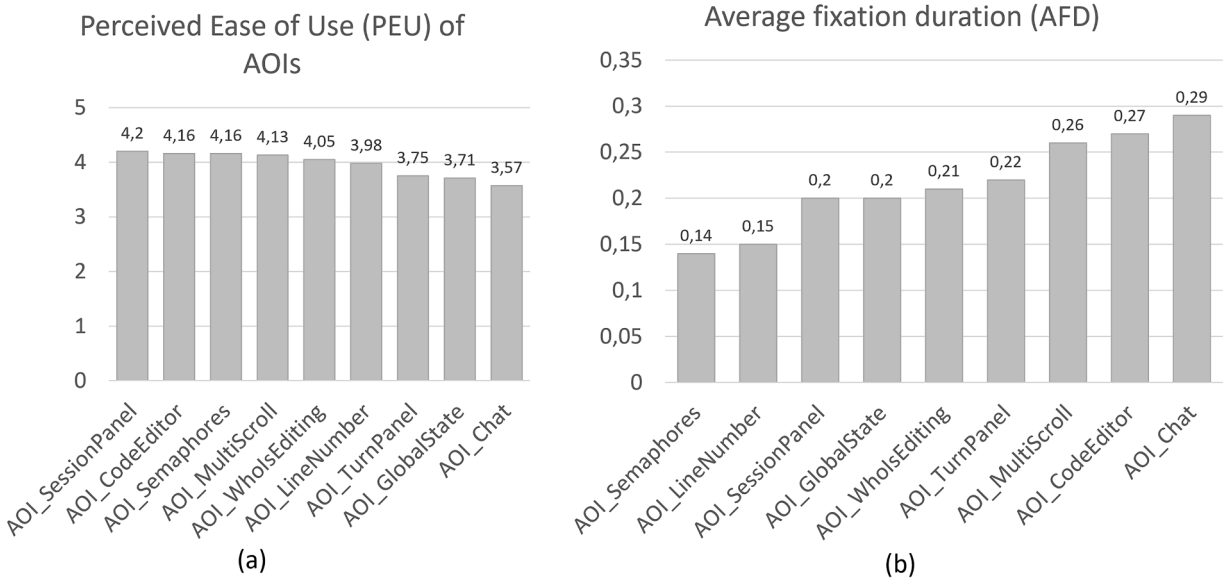

Fig. 8. (a) Ranking of AOIs from highest to lowest perceived ease of use (PEU) by participants after performing the collaborative programming activity with COLLECE. Result of the data analysis of all participants ($n = 86$). **(b)** Ranking of AOIs from least to most cognitive load (measured by the eye tracker metric "average fixation duration"). Result of the analysis of the data of the participants whose activity was recorded by the eye tracker device ($n = 33$).

Table 13
Average values of interaction count with the coordination support components, voting tools and chat, including the different types of messages supported (cmt: *chat message type*). Values per pair and per individual are shown.

	Per group [a]			Per participant [b]		
	[Min – Max]	Mean	Std. dev.	[Min – Max]	Mean	Std. dev.
Turn change count (TCC)	[1–10]	4.33	2.17	[1–10]	4.25	2.16
Compilation count (CC)	[0–12]	2.17	3.05	[0–12]	2.16	3.08
Execution count (EC)	[0–5]	0.59	1.17	[0–5]	0.56	1.16
Number of chat messages (NCM)	[1–86]	24.35	15.83	[0–45]	12.01	8.39
cmt1: "I think that…"	[0–83]	18.11	14.72	[0–43]	9.80	7.63
cmt2: "Why …"	[0–12]	1.30	2.06	[0–11]	0.71	1.48
cmt3: "I miss a …"	[0–5]	0.74	1.32	[0–4]	0.40	0.85
cmt4: "I see a mistake in …"	[0–4]	0.67	1.14	[0–4]	0.36	0.81
cmt5: "Revise the {}"	[0–2]	0.48	0.69	[0–2]	0.26	0.49
cmt6: "We must control …"	[0–6]	0.46	1.01	[0–3]	0.25	0.62
cmt7: "In the loop …"	[0–3]	0.15	0.56	[0–2]	0.08	0.35
cmt8: "Correct errors top-bottom."	[0–1]	0.02	0.15	[0–1]	0.01	0.11
cmt9: "Do we test more cases?"	[0–0]	0.00	0.00	[0–0]	0.00	0.00
cmt10: "This already works."	[0–2]	0.26	0.54	[0–2]	0.14	0.38

[a] Values for $n = 46$ groups, whose behavior was recorded by the automatic *logging* module.
[b] Values for $n = 85$ participants, whose behavior was recorded by the automatic *logging* module.

that, although statistically significant correlations have been obtained, in most cases, these correlations are weak.

RQ4. ***Is there any relationship between the support for the dimensions of Gutwin and Greenberg's framework (Who, What, and Where) and the quality of the product (program) obtained collaboratively?***

To answer the last research question (**RQ4**), related to the support for the dimensions of Gutwin and Greenberg's framework, participants' ratings of each of the three dimensions considered in this experiment (*Who, What* and *Where*) and their subdimensions were analyzed, as Fig. 9 resumes. Table 16 shows the mean values. As can be seen, the *Identity* subdimension is the most valued ($m = 4.16$, sd=0.83), followed by the *Artifact* ($m = 4.07$, sd=0.90) and *Changes* ($m = 4.02$, sd=0.86) dimensions. The worst-rated subdimension was *Authorship* ($m = 2.95$, sd=1.21), concerning which mechanisms are necessary to inform group members about who is doing what in the shared workspace.

On the other hand, Table 17 shows the correlations between the PU of the different elements of the COLLECE interface and the dimensions and subdimensions proposed in Gutwin and Greenberg's framework. As can be seen, there is consistency between the classification made by the authors in Table 6 regarding the support for the different dimensions in the following cases: code editor and multi-scroll bar (dimensions *Who, What,* and *Where*), in the case of the chat, and the floor control panel (*Who* and *What*). On the other hand, there is no total agreement for some dimensions in the rest of the interface elements.

Correlating the metrics provided by the eye tracker related to attention (in particular, the percentage of inspection time) and Gutwin and Greenberg's awareness dimensions, we obtain a negative correlation between the attention paid to the AOI indicating the global state in the group programming session and the dimensions *Who* (r=−0. 378, $p < 0.01$) and its subdimensions *Presence* (r=−0.335, $p < 0.05$) and *Identity* (r=−0.337, $p < 0.05$), as well as in the subdimension of *Artifact* (r=−0.292, $p < 0.05$) of dimension *What*. On the other hand, a correlation was found between the attention paid to the chat and the *Location* (*Where*) subdimension ($r = 0.365$, $p < 0.05$). As indicated in Table 5, the chat is considered to support the *Who* and *What* dimensions, however, considering the type of specific contributions that can be made through this communication tool when using sentence openers (for example: "*In the loop…*", "*Revise the {}*", "*I see a mistake in …*"), it can be seen that information is also being given about code portions, so it is also supporting the *Where* dimensions and, specifically, the *Location* subdimension.

A correlation was found between the rating given by the participants to the different subdimensions of awareness and the score obtained for the program developed collaboratively. A correlation, although weak, was obtained between obtaining a better-scored product and the *Artifact* subdimension ($r = 0.319$, $p < 0.01$). Considering the program quality indicators, significant correlations were obtained between having finished the program and scoring better on the *What* dimension ($r = 0.244$, $p < 0.04$) and the *Artifact* subdimension ($r = 0.299$, $p < 0.01$). On the contrary, a negative correlation was obtained between having completed the program and the rating given to the *Authorship* awareness

Table 14
Correlations between the score obtained and the interaction counts performed with the communication and coordination components. (a) Values for $n = 46$ groups, whose behavior was recorded by the automatic *logging* module. (b) Values for $n = 85$ participants, whose behavior was recorded by the automatic *logging* module.

Correlation analysis by groups [a]					
	EVAL-PROG	**Number of chat messages (NCM)**	**Turn change count (TCC)**	**Compilation count (CC)**	**Execution count (EC)**
EVAL-PROG	1.000				
Number of chat messages (NCM)	0.112	1.000			
Turn change count (TCC)	0.029	0.293*	1.000		
Compilation count (CC)	0.549**	0.090	−0.011	1.000	
Execution count (EC)	0.433**	0.076	0.117	0.704**	1.000
Correlation analysis by participant [b]					
	EVAL-PROG	**Number of chat messages (NCM)**	**Turn change count (TCC)**	**Compilation count (CC)**	**Execution count (EC)**
EVAL-PROG	1.000				
Number of chat messages (NCM)	0.014	1.000			
Turn change count (TCC)	0.006	0.278*	1.000		
Compilation count (CC)	0.577**	0.077	−0.042	1.000	
Execution count (EC)	0.459**	0.043	0.046	0.702**	1.000

Calculated statistic: Spearman's Rank Correlation Coefficients. The level of significance (alpha value):.
** The correlation is significant at the 0.01 level (bilateral).
* The correlation is significant at the 0.05 level (bilateral).

Table 15
Correlations between the quality criteria of the solution provided (EVAL-PROG components) and the interaction counts (by participant) performed with the communication and coordination components. Values for $n = 85$ participants, whose behavior was recorded by the automatic *logging* module.

	Completed program	Fit of implementation to algorithm	Providing the correct solution	Use of D&C	Correct use of indexes	Correct implementation of recursive calls	Correct definition of the base case	Solution efficiency
Number of chat messages (NCM)	0.108	0.075	−0.063	0.059	−0.123	0.002	0.091	0.049
Turn change count (TCC)	0.143	0.149	−0.014	0.064	−0.135	−0.008	−0.010	−0.060
Compilation count (CC)	0.559**	0.566**	0.271*	0.509**	.465**	0.457**	0.594**	0.576**
Execution count (EC)	0.410**	0.449**	0.267*	0.287**	.351**	0.320**	0.428**	0.446**
cmt1: "I think that…"	0.075	<0.001	−0.087	−0.015	−0.165	−0.053	−0.027	0.018
cmt2: "Why …"	0.049	.228*	0.045	0.241*	0.069	0.159	0.288**	0.125
cmt3: "I miss a …"	0.110	0.047	0.015	0.162	−0.084	0.013	0.129	0.017
cmt4: "I see a mistake in …"	0.098	.224*	0.138	0.196	−0.017	0.095	0.100	0.237*
cmt5: "Revise the {}"	0.045	−0.030	−0.162	−0.073	−0.195	−0.030	−0.149	−0.031
cmt6: "We must control …"	0.023	−0.014	−0.031	0.004	−0.178	−0.004	−0.035	0.025
cmt7: "In the loop …"	−0.067	−0.252*	−0.171	−0.216*	−0.161	−0.286**	−0.183	−0.112
cmt8: "Correct errors top-bottom."	0.106	0.020	−0.089	0.115	0.188	−0.084	0.040	−0.071
cmt9: "Do we test more cases?"	–	–	–	–	–	–	–	–
cmt10: "This already works."	0.222*	0.342**	0.129	0.282**	0.096	0.214*	0.271*	0.278*

Calculated statistic: Spearman's Rank Correlation Coefficients. The level of significance (alpha value):.
** The correlation is significant at the 0.01 level (bilateral).
* The correlation is significant at the 0.05 level (bilateral).

subdimension (r=−0.213, $p < 0.05$).

Regarding the relationship between the participants' profile and the rating given to the support for each of the dimensions and subdimensions of the awareness framework, it was found that those who rated their previous programming knowledge as higher scored the subdimension artifact (what) positively ($R = 0.236$, $P < 0.05$). On the other hand, those who reported greater experience in the use of collece also indicated that this system provides good support for the location (where) subdimension ($R = 0.230$, $P < 0.05$).

3.8. Strengths, weaknesses, and proposals for improvement

Finally, the responses given by the participants to the open-ended

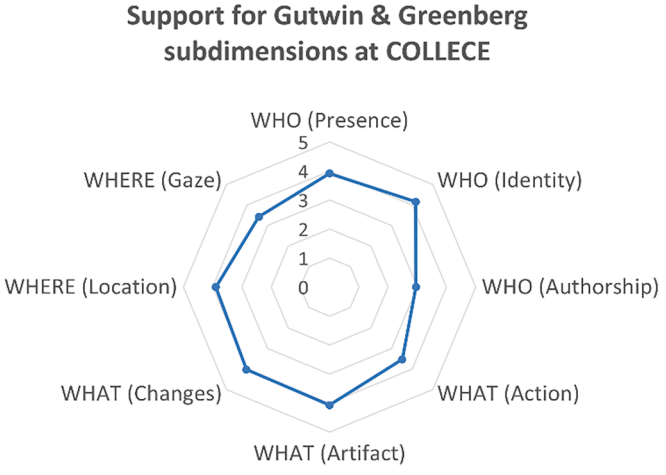

Fig. 9. Support for the awareness subdimensions of Gutwin and Greenberg's framework.

Table 16
COLLECE's support for the awareness dimensions and subdimensions of Gutwin and Greenberg's framework. Values for $n = 87$ participants.

Dim.	Subdimension	Ítem	Heuristic Evaluation [(a)]
Who	Presence	Is anyone in the workspace?	3.92 (0.85)
	Identity	Who is participating in the group programming activity? Who are they?	4.16 (0,83)
	Authorship	Who edited which part of the code?	2.95 (1.21)
What	Action	What is the peer doing? What are the activities they are currently doing?	3.52 (1.10)
	Artifact	What part of the code is the peer working on?	4.07 (0.90)
	Changes	What changes is the peer making to the code?	4.02 (0.86)
Where	Location	Where is the peer editing?	3.90 (1.00)
	Gaze	Where is the peer looking? Where in the code is his or her focus of attention?	3.43 (1.11)

[(a)] We show the mean and, in parentheses, the standard deviation.

questions were compiled. Regarding COLLECE's **strengths**, they highlighted the fact that it supports synchronous and distributed collaborative editing, as well as its ease of use. As for awareness support elements, many of the participants positively valued the fact that the line of code being edited by a colleague was highlighted at each moment, as well as the usefulness of the session panel and the multi-scroll bar.

Among the **negative** aspects of the system, most respondents indicated that the structured chat and sentence openers were not useful as they restricted communication, making it less fluid. They also lacked a more prominent notification (an audio or visual cue) to indicate turn requests and agreements, since some of them missed both situations

Table 17
Correlation between the rating of perceived usefulness (PU) in the post-test questionnaire and the ratings given to the support for the different dimensions of awareness.

	Perceived Usefulness (PU-Post-test)								
	Code Editor	Chat	Floor Control Panel	Semaphores	Session Panel	Multi-scroll bar	Who Is Editing	Line Number	Global State
HEA-WHO	0.360**	0.224*	0.360**	0.410**	0.187	0.243*	0.466**	0.299**	0.304**
HEA-WHAT	0.377**	0.252*	0.309**	0.384**	0.157	0.276**	0.398**	0.124	0.158
HEA-WHERE	0.297**	0.211	0.098	0.291**	0.198	0.250*	0.348**	0.396**	0.097
Who (Presence)	0.412**	0.164	0.362**	0.376**	0.128	.285**	0.327**	0.224*	0.226*
Who (Identity)	0.409**	0.135	0.265*	0.255*	0.214*	0.196	0.258*	0.101	0.019
Who (Authorship)	0.095	0.150	0.181	0.250*	0.118	0.151	0.388**	0.256*	0.308**
What (Action)	0.192	0.295**	0.304**	0.248*	0.112	0.230*	0.335**	0.155	0.061
What (Artifact)	0.478**	0.226*	0.281**	0.379**	0.096	0.215*	0.352**	0.094	0.080
What (Changes)	0.325**	0.075	0.104	0.293**	0.146	0.227*	0.252*	0.073	0.240*
Where (Location)	0.273*	0.109	0.002	0.234*	0.155	0.231*	0.226*	0.269*	0.052
Where (Gaze)	0.195	0.188	0.170	0.224*	0.154	0.152	0.328**	0.362**	0.093

Calculated statistic: Spearman's Rank Correlation Coefficients. The level of significance (alpha value):.
** The correlation is significant at the 0.01 level (bilateral).
* The correlation is significant at the 0.05 level (bilateral).

while programming.

Regarding **coordination**, some of the participants also pointed out that they did not see the necessity to reach consensus or to perform in group certain tasks, such as compilation. Collaborative coding, based on the assignment of editing turns, was well appreciated by some participants, while others considered it too strict, proposing to block code portions (lines or sets of lines) and not the whole code. They also felt that the appropriation of the turn by one of the members of the group would be problematic, proposing the inclusion of timers or the control of inactivity periods that would cause the turn to be automatically passed to the peer.

While some of the participants appreciated having all the functionalities available at once in the UI, others indicated that the UI was too complex and overloaded, proposing to modify its appearance to simplify it. They also considered it necessary to improve other features of the application, such as numbering the lines of code or the possibility of hiding or resizing components in the UI (in particular, the editing area and the chat). They also missed some features of professional IDEs, such as code highlighting or auto-completion functions (e.g., IntelliSense). Finally, and in relation to communication support, some participants pointed out the convenience of including additional communication channels based on audio and video.

4. Results and discussion

The main findings obtained in the experiment, organised around the research questions (RQ), are presented in a structured and summarised way in Table 18. For each RQ, the aspects studied as well as the results obtained are outlined.

We can state that the usefulness of the collaboration support mechanisms in COLLECE is high overall (**RQ1**), with certain components of the user interface standing out in terms of perceived usefulness and intention to use in the whole activity. Indeed, the fact that in the pre-test, subjects identified the same four UI components as both the most useful and those they had the greatest intention to use, provides confidence in a suitable UI design and the selection of the awareness widgets. As can be seen in the table of results (Table 18), it is concluded that the element for creating the shared artifact (the code editor) is the element considered the most useful and to which they devote the most attention during the group programming task, which makes sense. In addition to the editor, the communication support tools (chat), and the awareness components and mechanisms related to coordination (semaphores and multi-scroll bars) were also considered the most useful elements. With respect to the behaviour of programmers according to their experience, we have been able to detect that the collaborative code editor is a key component, since users with higher programming skills value the code editor more as a useful element (in line with this, the data reveal that the session panel and chat received little attention from these users). On the other hand, those with more experience with groupware systems report a high intention to use the code editor, perhaps because they were less afraid of the complexity of the dynamics of collaborative work itself due to their experience. This is also supported by the fact that they consulted fewer the coordination semaphores.

Despite the inherent complexity of combining a challenging task, programming, with synchronous distributed collaboration, the collaboration support mechanisms have proven to be easy to use (**RQ2**). In particular, the session panel, the code editor, and the semaphores stand out from the rest of the UI components. The first of these (the session panel) is a familiar component to students, as it is common in most existing synchronous communication tools. In addition to this UI component, the semaphore, an awareness mechanism based on the colour coding of this well-known traffic signal, is also considered easy to use. This element generates a low cognitive load for users since, with a quick visual scan, they can identify the information provided related to coordination and turn-taking. In contrast, the chat and the code editor, despite being considered the most useful elements, are the ones that impose the greatest cognitive load on the user. This result makes sense since their use implies reading and comprehending the content displayed in both components, which are the two most cognitively demanding tasks (Andrzejewska and Skawińska, 2020; Khomokhoana, 2020).

Regarding the user's fixation on a given UI component, the code editor, the chat, and the group scrollbar are close to double the fixation time of the semaphores and the line number. These low times can be due to different reasons: the visual signals of the traffic lights are very quick to perceive, as mentioned above, while the information provided by the line number is more the line change than the line number itself, except in the debugging phases where compilation errors need to be corrected. Precisely, it is evident that users with more experience in Java programming need to look less at the line number indicator.

Next, we are going to deal with question **RQ3**, which relates components of the COLLECE user interface to product quality. The first thing that is noticeable is the way in which the structured chat is used. The large number of times that the generic opener "*I think that…*" is used suggests that the use of chat in the freest possible way is preferred. The need to select the type of contribution to the chat was criticized by the participants in the experiment. This may be because they considered that it made synchronous communication slower, making it less fluid (Farnham et al., 2000). Even having detected some remarkable correlations, everything indicates that structured chat does not have a special interest in this field and could be discarded as a tool that seeks to improve the quality of the product obtained.

Table 18
Summary of the results obtained in the empirical study in response to the research questions (RQ) posed.

RQ	Studied aspect	Results
RQ1	UI components perceived as *most useful*	1. Code editor 2. Semaphores 3. Chat 4. Multi-scroll bar
	UI components visually *most attended*	1. Code editor 2. Chat 3. Turn panel 4. Session panel
RQ2	UI components *easiest to use*	1. Session panel 2. Code editor 3. Semaphores 4. Multi-scroll bar
	UI components imposing the highest *cognitive load*	1. Chat 2. Code editor 3. Multi-scroll bar 4. Turn panel
RQ3	Relationship between using certain components of the user interface and obtaining a better product	• Groups that compiled and executed the program more times obtained a higher score. • Greater use of more generic sentence introducers ("*I think that…*", "*Why*").
RQ4	Relationship between support for the dimensions of G&G's framework and the quality of the product obtained collaboratively	• Sub-dimensions best supported in COLLECE: - Identity (*Who*) - Artifact (*What*) - Changes (*What*)

As for the use of execution and compilation tools, the results indicate that the groups that made more intensive use of these tools achieved better scores. That is, in this case, and perhaps because the students are learning to program, the support of these tools is essential, in "trial-and-error" cycles, to reach the desired solution (Yamaguchi et al., 2022). These results may lead us to believe that work should be done to make it easy to compile and execute, since these options are necessary to achieve the desired result. This could imply not requiring consensus to carry out these actions, as has been discussed elsewhere. Finally, the correlations between communication and the number of editing turns suggest that a good collaborative approach extends to all the different tasks involved in the problem-solving activity: chatting, editing, etc.

Regarding the support that COLLECE provides for the dimensions of the Gutwin and Greenberg awareness support evaluation framework (**RQ4**), participants indicated that they felt that they were well informed about who was participating in the group programming activity. This information is provided primarily through the session panel and the multi-scroll bars, which were the elements rated as most useful. Also, the COLLECE interface allows the user to know at any time on which part of the shared artifact (source code) the partner is working (*Where*) and what changes he or she is making (*What*). There are several components that visualize such information, highlighting the multi-scroll bar and the code editor equipped with tele-cursors.As indicated in Table 6, the code editor and multi-scroll bar are clearly related to the dimensions *Who, What* and *Where*, whereas the floor control panel is related to *Who* and *What*. The discovery that has also emerged from the various data sources we have worked with is that the chat, which in principle would be related to *Who* and *What*, also has a relationship with *Where*, especially when line numbers are mentioned. This gives an idea of the usefulness of chat in this sense and of the possible integration of shortcuts to a particular line by clicking in the chat. On the other hand, the data reveal that those who positively rated the sub-dimensions *Presence* and *Identity* (*Who*) and *Artefact* (*What*) did not need to pay much attention to the global status indicator, as they were perfectly aware of what they were doing and with whom they were doing it through the activity itself and the corresponding UI components; on the contrary, looking at this global status was a help for those who were not so aware of the value of the who and what dimensions of the COLLECE's awareness.

Regarding the relationship between the support for the different awareness dimensions and the quality of the product obtained in the group activity, the correlations found between the quality and the *What* dimension, and in particular the *Artifact* subdimension, lead us to think that in this scenario the important thing is the work on the program, and it was not possible to detect that the support for the other dimensions could have benefited the final product. There are several arguments to be made about this, which in principle could be in detriment to what is stated in the rest of the paper. So, just because these UI elements do not affect quality does not mean that they are not really necessary. They are elements that have been shown to promote awareness support and therefore contribute to group work. However, for the quality of the product, there are other influential factors, such as the previous level of the participants, and all this leads to the fact that it is really the programming of the program itself -the *Artifact*- what receives greater importance in terms of the dimensions that affect quality.

The final and qualitative user feedback suggests that there is room for improvement in terms of the editing features of the code editor and the responsive nature of the user interface, but as stated, COLLECE is a learning and research tool and not a commercial or performance-seeking product. There is also room for improvement in the implementation of the small protocols for coordination, and it is clear that a structured chat is not required or does not contribute in any way to collaborative programming.

4.1. Implications for practice and transferability

This research work presents a variety of **implications for practice** and application, as follows:

1. At the ***technical level***, the results obtained allow *better-grounded redesign decisions* to be taken. Thus, based on the results obtained in the experiment, several *changes in the user interface* of COLLECE are proposed, such as replacing the structured chat with a conventional chat. Although the inclusion of structured chat in COLLECE was motivated by the objective of supporting more guided and organized group decision-making (van Dolen et al., 2007), as other authors also point out, its use is not well accepted by some users, who consider that its use negatively affects the natural flow of conversation (Farnham et al., 2000).

   In addition, it is planned to allow certain awareness support components to be displayed only on user demand, i.e., to support the so-called *awareness filters* (David and Borges, 2001). Their use would simplify the current COLLECE user interface, reducing *information overload*, which is identified as one of the problems raised by the inclusion of awareness mechanisms (Collazos et al., 2019).

   This capacity for adaptation and customization will also help to tackle the *interruptions* caused by receiving notifications and on-screen visualizations of information about others' actions, another common problem in groupware systems (Collazos et al., 2019). In this sense, the use of additional information channels (e.g., video and/or audio channels) (Metatla et al., 2018), which do not add visual load to the UI, can be of great help, enabling more fluid communication and coordination as well as a better redirection of attentional focus (Daly-Jones et al., 1998; Silva et al., 2020). Therefore, it is also envisaged to incorporate this improvement in the evolution of the COLLECE system.

   Although the proposed modifications and improvements are specific to the case of the evaluated system (COLLECE), the design decisions taken (basing textual communication on a conventional chat, the use of awareness filters and the inclusion of additional communication channels) can be extended to the design or redesign of other collaborative systems, either to support group programming or to perform other types of synchronous distributed tasks.

2. At a ***methodological level***, this work describes in detail how to carry out an evaluation of the communication, coordination, and awareness support of a collaborative programming tool by combining subjective and heuristic assessment techniques with objective information gathering techniques, such as eye tracking. Although the experiment carried out is particularized to the evaluation of a system that supports collaborative programming, the method and most of the measuring instruments employed could also be used to assess the support for collaboration in any groupware tool, framed both in the CSCW and CSCL fields.

   This experience, together with others previously conducted by the authors of this paper using this assessment methodology, has resulted in a set of *practical guidelines and recommendations*, which can be found in Molina et al. (2024). This other publication compiles some of the *good practices* to take into account when designing and carrying out experiments using the eye tracking technique, in the fields of Human-Computer Interaction and Educational Computing and with emphasis on how it helps to evaluate aspects related to human factors, usability and accessibility, although it does not deal with collaborative work neither awareness.

In terms of **transferability**, several dimensions can be distinguished: the eye tracking-based assessment approach, the domain (or type of task oriented to that domain), the level of awareness, and the metaphor or style of human-computer interaction of the system. The eye movement tracking method is based on eye fixation in AOI, but the approach taken can be extended to consider other eye tracking metrics based on social attention (Wohltjen and Wheatley, 2024), such as pupil size or eye blinks, to complement the data analysis. In addition to the programming domain, the evaluation method would be applicable to other tasks or domains that are based on the creation of texts using collaborative editors and, without major changes, to electronic spaces for the collaborative construction of graphic artifacts. In the latter case, including the possibility of linking eye actions with the creation of the objects of the model (artifact), which would increase the information to be analyzed (Molina et al., 2014). As for the type of awareness considered, the fact of using the eye tracking technique limits its use to the study of visual awareness mechanisms, which are available in the interface that users have in front of them and which they observe, even though other types of auditory awareness mechanisms, or tactile or body perception (other than the visual), can be studied through the other instruments of the method described (heuristic, qualitative evaluation, etc.). Finally, in terms of interaction style, the method is based on screen-based interfaces, but this also takes into account mobile apps or mixed reality devices, as technology has advanced and eye tracking sensors are now available that can be head-mounted for mobile devices or in the glasses used in virtual/augmented reality applications (Ban et al., 2022).

4.2. Limitations and threats to validity

The interpretation of the results and, consequently, the formulation of the research conclusions are subject to the following threats to the validity (Cook et al., 2002) of the study:

- **Internal validity**, which refers to the degree to which an experiment excludes alternative explanations for the results. The main threats associated with this issue are split according to participants, tasks, and measurement techniques.
  - *Participants:* The students might not have been sufficiently competent in the use of COLLECE because they only received a 45-minute seminar and did not have time to familiarize themselves with it, as Table 8 shows. Therefore, all participants received a short COLLECE user manual for reference during the experiment. Besides, the subjects might not have been properly motivated, which imposes another threat to the validity of our experiment. In this sense, students were not forced to participate in the experiment, and they received some extra points that were not required to pass the exam. Nevertheless, while watching the recorded user sessions, we did not encounter any unmotivated user behaviour.

    On the other hand, the *Hawthorne effect* (Chiesa and Hobbs, 2008), i.e., the phenomenon where individuals modify their behaviour when they are aware of being observed in a study, has been limited by anonymising the participants so that they would not feel judged. Finally, the use of subjective questionnaires (perception-based) may provide biased answers, mainly because the results are based upon the perceptions of the respondents. We have tried to limit this thread by tracking their eye movements on the screen to confirm their answers.

    However, some uncontrolled extraneous factors (Bhandari, 2022), such as personal feelings or circumstances, situational variables, or students' motivation, could affect participants' responses in the post-test questionnaire. But the control of these factors is beyond the scope of our research.
  - *Tasks.* One task-related threat is that the solutions were incorrectly rated. We mitigated this threat by anonymizing the solutions and employing an assessment rubric agreed upon by all evaluators (Table 4). The tasks might have been too difficult, which imposes another threat to validity. Table 8 also shows that most participants rated themselves as having regular programming experience but not in the *divide and conquer* technique, which might be insufficient for the proposed task. This limitation has been palliated by providing students with the pseudocode algorithm that solved the proposed task.
  - *Measurement techniques.* There are limitations and difficulties derived from the use of the eye tracking technique. First, only the information of one of the team members is tracked due to the lack of two *devices*, which would have allowed dual-eye tracking of the teamwork session (Dangelo and Schneider, 2021; Jermann et al., 2012).

    As described in Section 3.4, there was a significant *sample loss* (Fig. 4) due to the duration of the task, the use of stimuli of a dynamic nature, and the possible fatigue of the participants, who could modify their corporal position during the session. Although corrective cues were given before and during the task, non-reliable recordings were produced (low accuracy and precision) (Tobii, 2011), which had to be eliminated from the final analysis. Since this circumstance was foreseen, over-recruitment of participants was performed (Bojko, 2013).

    When performing the evaluation of a *dynamic stimulus* (a graphical user interface, as in the case of COLLECE), it was necessary to replay one by one the recordings of each of the participants whose visual behaviour was tracked. Since during the activity menus may be displayed, dialog boxes may appear, etc. that may hide the AOIs, it was necessary to pre-process and segment the recording information to obtain the final metrics.

    Regarding the analysis of the information gathered, it is also necessary to comment that there is still no consensus on the *terminology*, the name of the *metrics*, or their *interpretation* (Sharafi et al., 2015a). We have taken as a reference the classification of measures in attention and mental effort indicators of the most prominent authors in this field (Bojko, 2013; Poole and Ball, 2006). On the other hand, the use of subjective perception measures has made it possible to guide and contrast interpretations.

    On the other hand, it must be ensured that the *environmental settings* of the experiment are well controlled, e.g., light conditions that may affect the recordings (Molina et al., 2024).

    In order to control and alleviate the limitations derived from the use of this technique, the experiment was carried out following good practices for the design and conduct of eye tracking experiments (Bojko, 2013; Carter and Luke, 2020; Molina et al., 2024;

Pernice and Nielsen, 2009), specifically in the field of software engineering (Sharafi et al., 2020).

- **Construct validity**, i.e., the extent to which the measuring instruments support the interpretation they reflect. Some limitations to this aspect are related to the formulation of the items. With the aim of avoiding this limitation, part of the instrument used is based on validated frameworks (for example, the TAM-based questionnaire), and an attempt was made to use unambiguous vocabulary when translating items from English to Spanish. However, the reliability of the scale has not been calculated, which may limit the validity of the construct. This thread should be addressed in future studies.
- **External validity** is related to the generalization of the results to the population. To guarantee the external validation of empirical studies, it is recommendable to recruit representative participants from the target population. Since participants have been recruited voluntarily, the researcher does not have certainty that the sample is representative of the generality. Moreover, the obtained results should be generalized to real-life settings (*ecological validity*), for instance, while programming at home or in real programming labs. In this sense, although the experiment has been performed in a research laboratory, researchers were kept away from the students so that they could act freely without being observed. Finally, only one member of each group is tracked, which implies that the objective measures only represent half of the sample. With the aim of alleviating this limitation, the tracked subjects were chosen randomly.

5. Conclusions and future work

The group programming approaches have proven to be beneficial practices, both in professional and educational contexts. We have studied several systems supporting this approach to understand their structure and functionality, and, in parallel, we have proposed a classification scheme as a partial contribution of this work.

The analysis in such systems of what communication, coordination, and awareness support mechanisms allow for developing collaborative programming activities in a more effective and efficient way is a question of interest that has been scarcely addressed in previous literature. Accordingly, we have carried out an evaluation experiment using the COLLECE system, an application to support group programming that includes a rich set of components to support these collaboration and awareness features, gathering this way empirical insights to address this research challenge.

In contrast to the evaluations usually performed on this type of system, which are generally based on subjective opinions collected through interviews and questionnaires, the present evaluation approach combines several methods of inspection, inquiry, and testing. As an innovative aspect, eye tracking techniques are incorporated into the assessment with the aim of complementing the information gathered through the classical techniques. The use of eye tracking provides valuable information regarding the underlying cognitive processes of programmers, such as attention and mental effort, allowing researchers to better understand processes and intentions. Thus, the described experiment combines subjective-qualitative measures and objective-quantitative information (physiological, non-conscious, and collected in real-time) to perform a more comprehensive analysis of the group programming activity.

Based on four research questions, we have been able to obtain concrete results, among which we can mention the following:

- The UI components identified as most useful for the collaborative programming task are the semaphores of the coordination panels and the chat, leaving the code editor out of the picture.
- The UI components with the most visual attention are the chat and the turn panel (again, not considering the code editor).
- The UI components identified as easiest to use are the session panel and the coordination semaphores.
- The UI components with the least cognitive load are the semaphores and the number of the code line being edited.

Furthermore, the study carried out has served to identify strengths, weaknesses, and opportunities for improvement of the awareness support of the evaluated system, such as that free-text chat is preferable to a structured chat, that audio or more visual cues are welcome to indicate requests and agreements in the coordination tools, and that the low-level protocols for the coordination panels could be redesigned to be more effective.

The results also confirm that the proposed blended evaluation method is a promising approach to evaluating the quality of awareness and collaborative support in groupware programming systems. And we believe that the described experiment, in particular its design, methods, and instruments, may be useful for those who want to assess the coordination, communication, and awareness support mechanisms in other groupware systems and domains, providing insights and evidence for improving the design of collaborative systems and making decisions on what UI collaborative components should be considered to better support the users' goals in a specific situation.

As future work, it is planned to consider the results on the usefulness and cognitive load of the components supporting collaboration and awareness in COLLECE and some others to complete the implementation of an evolved version of this application, called COLLECE 2.0 (Lacave et al., 2019). Once these changes have been made, an evaluation of this new version will be carried out to contrast the outcome with the results of the present work. In these empirical evaluations, a greater number of participants will be involved, and other sources of physiological information will be used, such as electroencephalography (EEG) (Endres et al., 2023). In addition, we also plan the incorporation of additional measures that consider motivational, affective (Lacave et al., 2020) and other individual differences of programmers (gender, programmer style, etc.) (Navarro-Cota et al., 2024) or the influence of the formation and configuration of programming teams (Revelo et al., 2021) to better understand their performance in group programming tasks.

CRediT authorship contribution statement

Ana I. Molina: Writing – review & editing, Writing – original draft, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Crescencio Bravo:** Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Funding acquisition. **Jesús Gallardo:** Writing – review & editing, Validation, Methodology, Investigation. **Carmen Lacave:** Writing – review & editing, Methodology, Investigation, Formal analysis. **Miguel A. Redondo:** Writing – review & editing, Supervision, Methodology, Investigation, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank the University of Castilla-La Mancha, which financed the acquisition of the Pilot Plant "Usability Laboratory" and the eye tracking device, as well as the UCLM-Telefónica Chair of "Advanced Interaction Systems for Digital Education", for supporting the research carried out by the CHICO group with this specialized equipment.

This work has been developed and funded within the framework of the following research projects: CODIFICA project (Ref. PID2021-125122OB-100), funded by MCIU/AEI/10.13039/501100011033/ and by the European Regional Development Fund (ERDF), EU, "A way of making Europe" and FRAWA Project (Ref. SBPLY/21/180501/000244)

funded by the Junta de Comunidades de Castilla-La Mancha and the ERDF.

Data availability

The dataset generated is available at https://doi.org/10.5281/zenodo.11000862. The analyses are available from the corresponding author on reasonable request.

References

Adeliyi, A., Wermelinger, M., Kear, K., Rosewell, J., 2021. Investigating remote pair programming in part-time distance education. In: Proceedings of the United Kingdom and Ireland Computing Education Research Conference, pp. 1–7. https://doi.org/10.1145/3481282.3481290.

Afkinich, J.L., Blachman-Demner, D.R., 2020. Providing incentives to youth participants in research: a literature review. J. Empir. Res. Hum. Res. Ethics 15 (3), 202–215. https://doi.org/10.1177/1556264619892707.

Alkharusi, H., 2022. A descriptive analysis and interpretation of data from likert scales in educational and psychological research. Indian J. Psychol. Educ. 12 (2), 13–16.

Alsaqqa, S., Sawalha, S., Abdel-Nabi, H., 2020. Agile software development: methodologies and trends. Int. J. Interact. Mob. Technol. (IJIM) 14 (11), 246. https://doi.org/10.3991/ijim.v14i11.13269.

Andrzejewska, M., & Kotoniak, P. (2020). Development of program comprehension skills by novice programmers–longitudinal eye tracking studies. 19(4), 521–541. 10.15388/infedu.2020.23.

Andrzejewska, M., Skawińska, A., 2020. Examining students' intrinsic cognitive load during program comprehension–an eye tracking approach. In: Bittencourt, I.I., Cukurova, M., Muldner, K., Luckin, R., Millán, E. (Eds.), Proceedings of the International Conference on Artificial Intelligence in Education, 12164. Springer International Publishing. https://doi.org/10.1007/978-3-030-52240-7. Issue Icl.

Antunes, P., Herskovic, V., Ochoa, S.F., Pino, J.A., 2014. Reviewing the quality of awareness support in collaborative applications. J. Syst. Softw. 89 (1), 146–169. https://doi.org/10.1016/j.jss.2013.11.1078.

Arguedas, M., Daradoumis, T., Xhafa, F., 2016. Analyzing how emotion awareness influences students' motivation, engagement, self-regulation and learning outcome. Educ. Technol. Soc. 19 (2), 87–103.

Arroyo, Y., Molina, A.I., Lacave, C., Redondo, M.A., Ortega, M., 2018. The GreedEx experience: evolution of different versions for the learning of greedy algorithms. Comput. Appl. Eng. Educ. 26, 1306–1317. https://doi.org/10.1002/cae.22023.

Ayres, P., Lee, J.Y., Paas, F., van Merriënboer, J.J.G., 2021. The validity of physiological measures to identify differences in intrinsic cognitive load. Front. Psychol. 12 (September). https://doi.org/10.3389/fpsyg.2021.702538.

Ayres, P., Paas, F., 2012. Cognitive load theory: new directions and challenges. Appl. Cogn. Psychol. 26 (6), 827–832. https://doi.org/10.1002/acp.2882.

Baker, K., Greenberg, S., Gutwin, C., 2001. Heuristic evaluation of groupware based on the mechanics of collaboration. Eng. Hum. Comput. Interact. 2254 (2001), 123–139. https://doi.org/10.1007/3-540-45348-2_14.

Ban, S., Lee, Y.J., Kim, K.R., Kim, J.H., Yeo, W.H., 2022. Advances in materials, sensors, and integrated systems for monitoring eye movements. Biosensors 12 (11), 1039. https://doi.org/10.3390/bios12111039.

Beasley, Z.J., Johnson, A.R., 2022. The impact of remote pair programming in an upper-level CS course. In: Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education, 1, pp. 235–240. https://doi.org/10.1145/3502718.3524772. *Vol. 1*.

Berkman, M.İ., Karahoca, D., Karahoca, A., 2018. A measurement and structural model for usability evaluation of shared workspace groupware. Int. J. Hum. Comput. Interact. 34 (1), 35–56. https://doi.org/10.1080/10447318.2017.1326578.

Bhandari, P., 2022. Extraneous Variables, Examples, Types & Controls. Scribbr. https://www.scribbr.com/methodology/extraneous-variables/.

Bojko, A., 2013. Eye Tracking the User Experience: A Practical Guide to Research. Rosenfeld Media.

Boyer, K.E., Wight, A.A.D., Taylor Fondren, R., Vouk, M.A., Lester, J.C., Dwight, A.A., Fondren, R.T., Vouk, M.A., Lester, J.C., 2008. A development environment for distributed synchronous collaborative programming. In: Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education, pp. 158–162. https://doi.org/10.1145/1384271.1384315.

Bravo, C., Duque, R., Gallardo, J., 2013. A groupware system to support collaborative programming: design and experiences. J. Syst. Softw. 86 (7), 1759–1771. https://doi.org/10.1016/j.jss.2012.08.039.

Bravo, C., Redondo, M.A., Verdejo, M.F., Ortega, M., 2008. A framework for process–solution analysis in collaborative learning environments. Int. J. Hum. Comput. Stud. 66 (11), 812–832. https://doi.org/10.1016/j.ijhcs.2008.08.003.

Burney, S.A., Ali, S.A., Ejaz, A., Siddiqui, F.A., 2017. Discovering the correlation between technology acceptance model and usability. IJCSNS Int. J. Comput. Sci. Netw. Secur. 17 (11), 53–61. http://paper.ijcsns.org/07_book/201711/20171107.pdf.

Carlisle, M.C., Wilson, T.A., Humphries, J.W., Hadfield, S.M., 2005. RAPTOR: a visual programming environment for teaching algorithmic problem solving. ACM SIGCSE Bull. 37 (1), 176–180.

Carter, B.T., Luke, S.G., 2020. Best practices in eye tracking research. Int. J. Psychophysiol. 155 (June), 49–62. https://doi.org/10.1016/j.ijpsycho.2020.05.010.

Celepkolu, M., Boyer, K.E., 2018. Thematic analysis of students' reflections on pair programming in CS1. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE 2018, pp. 771–776. https://doi.org/10.1145/3159450.3159516.

Cepero, M.T., Montané-Jiménez, L.G., Toledo-Toledo, G., Benítez-Guerrero, E.I., Mezura-Godoy, C., 2021. Heuristics for awareness support in groupware systems. DYNA New Technol. 8 (1), 11. https://doi.org/10.6036/NT9980 [p.]-[11 p.].

Chiesa, M., Hobbs, S., 2008. Making sense of social research: how useful is the Hawthorne Effect? Eur. J. Soc. Psychol. 38 (1), 67–74.

Chorfi, A., Hedjazi, D., Aouag, S., Boubiche, D., 2022. Problem-based collaborative learning groupware to improve computer programming skills. Behav. Inf. Technol. 41 (1), 139–158. https://doi.org/10.1080/0144929X.2020.1795263.

Ciolfi, L., Lewkowicz, M., Schmidt, K., 2023. CSCW: history, core issues, and approaches in computer-supported cooperative work. In: Vanderdonckt, J., Palanque, P., Winckler, M. (Eds.), Handbook of Human Computer Interaction. Springer.

Cohen, J., 1992. Quantitative methods in psychology: a power primer. Psychol. Bull. 112 (1), 155–159.

Collazos, C.A., Fardoun, H., AlSekait, D., Pereira, C.S., Moreira, F., 2021. Designing online platforms supporting emotions and awareness. Electronics 10 (3), 251. https://doi.org/10.3390/electronics10030251.

Collazos, C.A., Guerrero, L., Pino, J., 2003. Knowledge construction awareness. J. Stud. Centered Learn. 2, 77–86.

Collazos, C.A., Gutiérrez, F.L., Gallardo, J., Ortega, M., Fardoun, H.M., Molina, A.I., 2019. Descriptive theory of awareness for groupware development. J. Ambient Intell. Humaniz. Comput. 10 (12), 4789–4818. https://doi.org/10.1007/s12652-018-1165-9.

Cook, T.D., Campbell, D.T., Shadish, W., 2002. Experimental and Quasi-Experimental Designs for Generalized Causal Inference. Houghton Mifflin, Boston, MA.

Coskun, A., Cagiltay, K., 2022. A systematic review of eye-tracking-based research on animated multimedia learning. J. Comput. Assist. Learn. 38 (2), 581–598. https://doi.org/10.1111/jcal.12629.

D'Angelo, S., Begel, A., 2017. Improving communication between pair programmers using shared gaze awareness. In: Proceedings of the Conference on Human Factors in Computing Systems, pp. 6245–6255. https://doi.org/10.1145/3025453.3025573.

da Silva Estácio, B.J., Prikladnicki, R., 2015. Distributed pair programming: a systematic literature review. Inf. Softw. Technol. 63, 1–10. https://doi.org/10.1016/j.infsof.2015.02.011.

Daly-Jones, O., Monk, A., Watts, L., 1998. Some advantages of video conferencing over high-quality audio conferencing: fluency and awareness of attentional focus. Int. J. Hum. Comput. Stud. 49 (1), 21–58. https://doi.org/10.1006/ijhc.1998.0195.

Dangelo, S., Schneider, B., 2021. Shared gaze visualizations in collaborative interactions: past, present and future. Interact. Comput. 33 (2), 115–133. https://doi.org/10.1093/iwcomp/iwab015.

David, J.M.N., Borges, M.R.S., 2001. Selectivity of awareness components in asynchronous CSCW environments. In: Proceedings of the Seventh International Workshop on Groupware. CRIWG 2001, pp. 115–124. https://doi.org/10.1109/CRIWG.2001.951824.

Davis, F.D., 1993. User acceptance of information technology: system characteristics, user perceptions and behavioral impacts. Int. J. Man Mach. Stud. 38 (3), 475–487. https://doi.org/10.1006/imms.1993.1022.

Davis, F.D., Granić, A., Marangunić, N., 2024. The technology acceptance model 30 years of TAM. In: SpringerBriefs in Human-Computer Interaction, 1. Springer.

Do Espirito Santo, D.A., De Oliveira, B.R., Hadad, F., Silva, F., 2018. Quality assessment of awareness support in agile collaborative tools. In: Proceedings of the 44th Latin American Computing Conference, CLEI 2018, pp. 21–30. https://doi.org/10.1109/CLEI.2018.00013.

Dourish, P., Bellotti, V., 1992. Awareness and coordination in shared workspaces. In: Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work - CSCW '92, pp. 107–114. https://doi.org/10.1145/143457.143468.

Duckert, M., Bjørn, P., 2024. Revisiting Grudin's eight challenges for developers of groupware technologies 30 years later. I-Com 1–25. https://doi.org/10.1515/icom-2023-0039.

Duque, R., Bravo, C., 2008. Supporting distributed pair programming with the COLLECE Groupware System: an empirical study. In: Proceedings of the International Conference on Agile Processes and Extreme Programming in Software Engineering, pp. 232–233.

Duque, R., Gómez-Pérez, D., Nieto-Reyes, A., Bravo, C., 2015. Analyzing collaboration and interaction in learning environments to form learner groups. Comput. Hum. Behav. 47, 42–49. https://doi.org/10.1016/j.chb.2014.07.012.

Ellis, C.A., Gibbs, S.J., Rein, G., 1991. Groupware: some issues and experiences. Commun. ACM 34 (1), 39–58. https://doi.org/10.1145/99977.99987.

Endres, M., Brechmann, A., Sharif, B., Weimer, W., & Siegmund, J. (2023). Foundations for a new perspective of understanding programming: *Vol. Report fro* (Issue 10).

Estler, H.C., Nordio, M., Furia, C.A., Meyer, B., 2013. Unifying configuration management with merge conflict detection and awareness systems. In: Proceedings of the 22nd Australian Software Engineering Conference, pp. 201–210. https://doi.org/10.1109/ASWEC.2013.32.

Fan, H., Sun, C., 2012. Achieving integrated consistency maintenance and awareness in real-time collaborative programming environments: the CoEclipse approach. In: Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 94–101. https://doi.org/10.1109/CSCWD.2012.6221803.

Farnham, S., Chesley, H.R., McGhee, D.E., Kawal, R., Landau, J., 2000. Structured online interactions: improving the decision-making of small discussion groups. In: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work, pp. 299–308. https://doi.org/10.1145/358916.359001.

Frías, S.G.D., Mezura-Godoy, C., Benítez-Guerrero, E., 2019. FrUtEG: a conceptual framework for utility evaluation in groupware. In: Proceedings of the IX Latin American Conference on Human Computer Interaction, pp. 1–8.
Gallardo, J., Bravo, C., Molina, A.I., 2018. A framework for the descriptive specification of awareness support in multimodal user interfaces for collaborative activities. J. Multimodal User Interfaces 12 (2), 145–159. https://doi.org/10.1007/s12193-017-0255-x.
Geszten, D., Hámornik, B.P., Hercegfi, K., 2021. Empirical study of team usability testing: a laboratory experiment. Cogn. Technol. Work 23 (4), 755–769. https://doi.org/10.1007/s10111-020-00647-8.
Geszten, D., Hámornik, B.P., Hercegfi, K., 2024. Team usability testing: development and validation of a groupware usability evaluation method. Cogn. Technol. Work, 0123456789. https://doi.org/10.1007/s10111-024-00759-5.
Granić, A., Marangunić, N., 2019. Technology acceptance model in educational context: a systematic literature review. Br. J. Educ. Technol. 50 (5), 2572–2593. https://doi.org/10.1111/bjet.12864.
Grant, R., Sugarman, J., 2004. Ethics in human subjects research: do incentives matter? J. Med. Philos. 29 (6), 717–738. https://doi.org/10.1080/03605310490883046.
Greif, I., 1988. Computer-Supported Cooperative Work: A Book of Readings. Morgan Kaufmann. https://dl.acm.org/citation.cfm?id=49504.
Gross, T., 2013. Supporting effortless coordination: 25 years of awareness research. Comput. Support. Coop. Work (CSCW) 22 (4–6), 425–474. https://doi.org/10.1007/s10606-013-9190-x.
Grudin, J., 1994. Computer-supported cooperative work: history and focus. Computer 27 (5), 19–26. https://doi.org/10.1109/2.291294.
Gutwin, C., Greenberg, S., 2000. The mechanics of collaboration: developing low cost usability evaluation methods for shared workspaces. In: Proceedings of the IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2000), pp. 98–103. https://doi.org/10.1109/ENABL.2000.883711.
Gutwin, C., Greenberg, S., 2002. A descriptive framework of workspace awareness for real-time groupware. Comput. Support. Coop. Work 11 (3–4), 411–446. https://doi.org/10.1023/A:1021271517844. *11: 411–446, 2002. 2002 Kluwer Academic Publishers. Printed in the Netherlands.*.
Gutwin, C., Greenberg, S., Roseman, M., 1996. Workspace awareness in real-time distributed groupware: framework, widgets, and evaluation. In: Proceedings of the HCI '96, pp. 281–298. https://doi.org/10.1145/257089.257286.
Halpern, S.D., Chowdhury, M., Bayes, B., Cooney, E., Hitsman, B.L., Schnoll, R.A., Lubitz, S.F., Reyes, C., Patel, M.S., Greysen, S.R., Mercede, A., Reale, C., Barg, F.K., Volpp, K.G., Karlawish, J., Stephens-Shields, A.J., 2021. Effectiveness and ethics of incentives for research participation. JAMA Intern. Med. 181 (11), 1479. https://doi.org/10.1001/jamainternmed.2021.5450.
Harpe, S.E., 2015. How to analyze Likert and other rating scale data. Curr. Pharm. Teach. Learn. 7 (6), 836–850. https://doi.org/10.1016/j.cptl.2015.08.001.
Hawlitschek, A., Berndt, S., Schulz, S., 2022. Empirical research on pair programming in higher education: a literature review. Comput. Sci. Educ. 1–57. https://doi.org/10.1080/08993408.2022.2039504.
Huang, Y., Leach, K., Sharafi, Z., McKay, N., Santander, T., & Weimer, W. (2020). Biases and differences in code review using medical imaging and eye-tracking: genders, humans, and machines. 456–468. 10.1145/3368089.3409681.
Hundhausen, C.D., Carter, A.S., 2014. Supporting social interactions and awareness in educational programming environments. In: Proceedings of the 5th Workshop on Evaluation and Usability of Programming Languages and Tools, pp. 55–56. https://doi.org/10.1145/2688204.2688215.
Hung, J.C., Wang, C.C., 2021. The influence of cognitive styles and gender on visual behavior during program debugging: a virtual reality eye tracker study. Hum. Centric Comput. Inf. Sci. 11 (22).
Jermann, P., Gergle, D., Bednarik, R., Brennan, S., 2012. DUET 2012 : dual eye tracking in CSCW. In: Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work Companion, CSCW '12, pp. 23–24. https://doi.org/10.1145/2141512.2141525.
Jermann, P., Nüssli, M.A., 2012. Effects of sharing text selections on gaze cross-recurrence and interaction quality in a pair programming task. In: Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work, pp. 1125–1134. https://doi.org/10.1145/2145204.2145371.
Jiménez, J.A., Collazos, C.A., Ortega, M., 2022. CodES: herramienta de visualización para desarrollo de pensamiento algorítmico. Campus Virtuales 11 (1), 21. https://doi.org/10.54988/cv.2022.1.809.
Jurado, F., Molina, A.I., Redondo, M.A., Ortega, M., 2013. Cole-programming: shaping collaborative learning support in eclipse. Rev. Iberoam. Tecnol. Del Aprendiz. (IEEE-RITA) 8 (4), 153–162. https://doi.org/10.1109/RITA.2013.2284953.
Just, M.A., Carpenter, P.A., 1980. A theory of reading: from eye fixations to comprehension. Psychol. Rev. 87 (4), 329.
Katona, J., Kovari, A., Heldal, I., Helgesen, C., Costescu, C., Rosan, A., Hathazi, A., Thill, S., Demeter, R., 2019. Recording eye-tracking parameters during a program source-code debugging example. In: Proceedings of the 10th IEEE International Conference on Cognitive Infocommunications, CogInfoCom 2019, pp. 335–338. https://doi.org/10.1109/CogInfoCom47531.2019.9089941.
Khomokhoana, P.J., 2020. Source Code Comprehension: Decoding the Cognitive Challenges of Novice Programmers. University of the Free State.
Lacave, C., Garcia, M.A., Molina, A.I., Sanchez, S., Redondo, M.A., Ortega, M., 2019. COLLECE-2.0: a real-time collaborative programming system on Eclipse. In: Proceedings of the International Symposium on Computers in Education (SIIE), pp. 1–6. https://doi.org/10.1109/SIIE48397.2019.8970132.
Lacave, C., Molina, A.I., 2021. The impact of COVID-19 in collaborative programming. Understanding the needs of undergraduate computer science students. Electronics 10 (14), 1728. https://doi.org/10.3390/electronics10141728.
Lacave, C., Velázquez-Iturbide, J.Á., Paredes-Velasco, M., Molina, A.I., 2020. Analyzing the influence of a visualization system on students' emotions: an empirical case study. Comput. Educ. 149, 103817. https://doi.org/10.1016/j.compedu.2020.103817.
Li, X., Liu, W., Wang, W., Zhong, J., Yu, M., 2019. Assessing students' behavior in error finding programming tests: an eye-tracking based approach. In: Proceedings of the IEEE International Conference on Engineering, Technology and Education, TALE 2019. https://doi.org/10.1109/TALE48000.2019.9225906.
Lin, Y.T., Wu, C.C., Hou, T.Y., Lin, Y.C., Yang, F.Y., Chang, C.H., 2016. Tracking students' cognitive processes during program debugging-an eye-movement approach. IEEE Trans. Ed. 59 (3), 175–186. https://doi.org/10.1109/TE.2015.2487341.
Liu, L., Liu, W., Li, X., Wang, W., Cheng, W., 2020. Eye-tracking based performance analysis in error finding programming test. In: Proceedings of the 15th International Conference on Computer Science and Education, ICCSE 2020, ICCSE, pp. 477–482. https://doi.org/10.1109/ICCSE49874.2020.9201882.
Lopez, G., Guerrero, L.A., 2017. Awareness supporting technologies used in collaborative systems. In: Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, pp. 808–820. https://doi.org/10.1145/2998181.2998281.
Lund, K., Baker, M., Baron, M., 1996. Modelling dialogue and beliefs as a basis for generating guidance in a CSCL environment. In: Proceedings of the Intelligent Tutoring Systems: Third International Conference, ITS'96. Montréal, Canada. Springer, pp. 206–214. https://doi.org/10.1007/3-540-61327-7_117. June 12–14, 1996 Proceedings.
Mantau, M.J., Benitti, F.B.V., 2022a. Towards an awareness taxonomy. In: Proceedings of the IEEE 25th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2022, pp. 495–500. https://doi.org/10.1109/CSCWD54268.2022.9776129.
Mantau, M.J., Benitti, F.B.V., 2022b. Awareness support in collaborative system: reviewing last 10 years of CSCW research. In: Proceedings of the IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 564–569. https://doi.org/10.1109/CSCWD54268.2022.9776091.
Mantau, M.J., Benitti, F.B.V., 2024. The awareness assessment model: measuring awareness and collaboration support over participant's perspective. Univers. Access Inf. Soc. 0123456789, 30–43. https://doi.org/10.1007/s10209-024-01110-5.
Mantau, M.J., Benitti, F.B.V., 2023. The awareness assessment model: measuring the awareness and collaboration support over the participant's perspective. In: Anais Do XVIII Simpósio Brasileiro de Sistemas Colaborativos, pp. 30–43. https://doi.org/10.5753/sbsc.2023.229064.
Metatla, O., Bryan-Kinns, N., Stockman, T., 2018. "I hear you": understanding awareness information exchange in an audio-only workspace. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, pp. 1–13. https://doi.org/10.1145/3173574.3174120.
Mitaritonna, A., Abasolo, M.J., Montero, F., 2019. Situational awareness through augmented reality: 3D-SA model to relate requirements, design and evaluation. In: Proceedings of the International Conference on Virtual Reality and Visualization, ICVRV 2019, pp. 227–232. https://doi.org/10.1109/ICVRV47840.2019.00054.
Molina, A.I., Arroyo, Y., Lacave, C., Redondo, M.A., Bravo, C., Ortega, M., 2024. Eye tracking-based evaluation of accessible and usable interactive systems: tool set of guidelines and methodological issues. Univers. Access Inf. Soc. 0123456789. https://doi.org/10.1007/s10209-023-01083-x.
Molina, A.I., Gallardo, J., Redondo, M.A., Bravo, C., 2015. Assessing the awareness mechanisms of a collaborative programming support system. DYNA 82 (193), 212–222. https://doi.org/10.15446/dyna.v82n193.53497.
Molina, A.I., Redondo, M.A., Ortega, M., Lacave, C., 2014. Evaluating a graphical notation for modeling collaborative learning activities: a family of experiments. Sci. Comput. Program 88, 54–81. https://doi.org/10.1016/j.scico.2014.02.019.
Natsu, H., Favela, J., Morán, A.L., Decouchant, D., Martinez-Enriquez, A.M., 2003. Distributed pair programming on the Web. In: Proceedings of the Fourth Mexican International Conference on Computer Science, 2003. ENC 2003, pp. 81–88.
Navarro-Cota, C., Molina, A.I., Redondo, M.A., Lacave, C., 2024. Individual differences in computer programming: a systematic review. Behav. Inf. Technol. 1–19. https://doi.org/10.1080/0144929X.2024.2317377.
Nielsen, J., Molich, R., 1990. Heuristic evaluation of user interfaces. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 249–256.
Nielsen, J., Pernice, K., 2010. Eyetracking Web Usability. New Riders.
Obaidellah, U., Al Haek, M., Cheng, P.C.H., 2018. A survey on the usage of eye-tracking in computer programming. ACM Comput. Surv. 51 (1). https://doi.org/10.1145/3145904.
Obaidellah, U., Haek, M.A., 2018. Evaluating gender difference on algorithmic problems using eye-tracker. In: Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications, pp. 1–8. https://doi.org/10.1145/3204493.3204537.
Ortega, M., Redondo, M.Á., Bravo, C., Molina, A.I., Lacave, C., Arroyo, Y., Sánchez-Sobrino, S., Navarro, Ó., Navarro, C.X., Gallardo, J., 2019. CHICO 2019 (Computer–human interaction and collaboration), UCLM. Inform. Educ. Comun. 30.
Papavlasopoulou, S., Giannakos, M.N., Sharma, K., Jaccheri, L., 2017. Using eye-tracking to unveil differences between kids and teens in coding activities. In: Proceedings of the 2017 ACM Conference on Interaction Design and Children, IDC 2017, pp. 171–181. https://doi.org/10.1145/3078072.3079740. December.
Peitek, N., Siegmund, J., Apel, S., 2020. What drives the reading order of programmers? An eye tracking study. In: Proceedings of the IEEE International Conference on Program Comprehension, pp. 342–353. https://doi.org/10.1145/3387904.3389279.

Pernice, K., & Nielsen, J. (2009). How to conduct eyetracking studies (Issue August). http://www.useit.com/eyetracking/methodology/.
Pietinen, S., Bednarik, R., Glotova, T., Tenhunen, V., Tukiainen, M., 2008. A method to study visual attention aspects of collaboration: eye-tracking pair programmers simultaneously. In: Proceedings of the Eye Tracking Research and Applications Symposium (ETRA), January, p. 39. https://doi.org/10.1145/1344471.1344480.
Pietinen, S., Bednarik, R., Tukiainen, M., 2010. Shared visual attention in collaborative programming. In: Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, pp. 21–24. https://doi.org/10.1145/1833310.1833314.
Pinelle, D., Gutwin, C., 2000a. A review of groupware evaluations. In: Proceedings of the IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 86–91.
Pinelle, D., Gutwin, C., 2000b. Groupware evaluations. Techniques 86–91.
Pinelle, D., Gutwin, C., 2002. Groupware walkthrough: adding context to groupware usability evaluation. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 455–462.
Pokhrel, S., Chhetri, R., 2021. A literature review on impact of COVID-19 pandemic on teaching and learning. High. Educ. Future 8 (1), 133–141. https://doi.org/10.1177/2347631120983481.
Poole, A., Ball, L.J., 2006. Encyclopedia of human computer interaction. In: Ghaoui, C. (Ed.), Encyclopedia of Human-Computer Interaction. IGI Global. https://doi.org/10.4018/978-1-59140-562-7.
Prechelt, L., Beecher, K., 2011. Four generic issues for tools-as-plugins illustrated by the distributed editor Saros. In: Proceedings of the 1st Workshop on Developing Tools as Plug-Ins, May 2011, pp. 9–11. https://doi.org/10.1145/1984708.1984712.
Revelo, O., Collazos, C.A., Redondo, M.A., 2021. Automatic group organization for collaborative learning applying genetic algorithm techniques and the big five model. Mathematics 9 (13), 1578. https://doi.org/10.3390/math9131578.
Revelo, O., Collazos, C., Jiménez, J., 2018. El trabajo colaborativo como estrategia didáctica para la enseñanza/aprendizaje de la programación: una revisión sistemática de la literatura. TecnoLógicas 21 (41), 123–7799. http://www.scielo.org.co/pdf/teclo/v21n41/v21n41a08.pdf%0A.http://www.scielo.org.co/scielo.php?pid=S0123-77992018000100008&script=sci_arttext&tlng=es%0Ahttp://www.scielo.org.co/pdf/teclo/v21n41/v21n41a08.pdf.
Richter, A., Koch, M., Prilla, M., 2024. CSCW–past, present and future. I-Com 1–5. https://doi.org/10.1515/icom-2024-0023.
Salomón, S., Duque, R., Montaña, J.L., Tenés, L., 2019. A method for analyzing the quality-in-use in collaborative contexts. In: Proceedings of the XX International Conference on Human Computer Interaction, pp. 1–8. https://doi.org/10.1145/3335595.3335633.
Schez-Sobrino, S., García, M.Á., Lacave, C., Molina, A.I., Glez-Morcillo, C., Vallejo, D., Redondo, M.Á., 2021. A modern approach to supporting program visualization: from a 2D notation to 3D representations using augmented reality. Multimed. Tools Appl. 80 (1), 543–574. https://doi.org/10.1007/s11042-020-09611-0.
Schmidt, K., 2002. The problem with `Awareness': introductory remarks on `awareness in CSCW. Comput. Support. Coop. Work (CSCW) 11 (3–4), 285–298. https://doi.org/10.1023/A:1021272909573.
Schmidt, K., Randall, D., 2016. Preface to the special issue on 'reconsidering "awareness" in CSCW. Comput. Support. Coop. Work (CSCW) 25 (4–5), 229–233. https://doi.org/10.1007/s10606-016-9257-6.
Schnaubert, L., Bodemer, D., 2022. Group awareness and regulation in computer-supported collaborative learning. Int. J. Comput. Support. Collab. Learn. 17 (1), 11–38. https://doi.org/10.1007/s11412-022-09361-1.
Sharafi, Z., Shaffer, T., Sharif, B., Gueheneuc, Y.G., 2015a. Eye-tracking metrics in software engineering. In: Proceedings of the Asia-Pacific Software Engineering Conference (APSEC), 2016-May, pp. 96–103. https://doi.org/10.1109/APSEC.2015.53.
Sharafi, Z., Sharif, B., Guéhéneuc, Y.G., Begel, A., Bednarik, R., Crosby, M., 2020. A practical guide on conducting eye tracking studies in software engineering. Empir. Softw. Eng. 25 (5), 3128–3174. https://doi.org/10.1007/s10664-020-09829-4.
Sharafi, Z., Soh, Z., Guéhéneuc, Y.G., 2015b. A systematic literature review on the usage of eye-tracking in software engineering. Inf. Softw. Technol. 67, 79–107. https://doi.org/10.1016/j.infsof.2015.06.008.
Sharma, K., Jermann, P., Nüssli, M.A., Dillenbourg, P., 2013. Understanding collaborative program comprehension: interlacing gaze and dialogues. In: Proceedings of the Computer-Supported Collaborative Learning Conference, CSCL, 1, pp. 430–437.
Silva, L., Mendes, A.J., Gomes, A., 2020. Computer-supported collaborative learning in programming education: a systematic literature review. In: Proceedings of the IEEE Global Engineering Education Conference (EDUCON), pp. 1086–1095. https://doi.org/10.1109/EDUCON45650.2020.9125237.
Stein, R., Brennan, S.E., 2004. Another person's eye gaze as a cue in solving programming problems. In: Proceedings of the Sixth International Conference on Multimodal Interfaces, ICMI'04, pp. 9–15. https://doi.org/10.1145/1027933.1027936.
Stone, D., Jarrett, C., Woodroffe, M., Minocha, S., 2005. User Interface Design and Evaluation. Elsevier.
Taherdoost, H., 2018. A review of technology acceptance and adoption models and theories. Procedia Manuf. 22, 960–967. https://doi.org/10.1016/j.promfg.2018.03.137.
Tam, J., Greenberg, S., 2006. A framework for asynchronous change awareness in collaborative documents and workspaces. Int. J. Hum. Comput. Stud. 64 (7), 583–598. https://doi.org/10.1016/j.ijhcs.2006.02.004.
Tobii. (2011). Accuracy and precision. Test report, Tobii T60 XL Eyetracker. http://www.tobii.com/.
Tobii Pro, 2016. Tobii Studio User's Manual. Tobii AB. http://www.tobiipro.com/siteassets/tobii-pro/user-manuals/tobii-pro-studio-user-manual.pdf.
Tsai, C.W., Lin, M.Y.C., Cheng, Y., Lee, L., Chyr, W., Lin, C.H., Lin, J., Tsai, M, 2023. The effects of online peer-facilitated learning and distributed pair programming on students' learning. Comput. Educ., 104849 https://doi.org/10.1016/j.compedu.2023.104849. *In press.*
Turenko, V., Baltulionis, S., Vasiljevas, M., Damaševičius, R., 2019. Analysing program source code reading skills with eye tracking technology. In: Proceedings of the CEUR Workshop, 2470, pp. 33–37.
van Dolen, W.M., Dabholkar, P.A., de Ruyter, K., 2007. Satisfaction with online commercial group chat: the influence of perceived technology attributes, chat group characteristics, and advisor communication style. J. Retail. 83 (3), 339–358. https://doi.org/10.1016/j.jretai.2007.03.004.
Villamor, M.M., Rodrigo, M.M.T., 2017. Characterizing collaboration in the pair program tracing and debugging eye-tracking experiment: a preliminary analysis. In: Proceedings of the 10th International Conference on Educational Data Mining, EDM 2017, pp. 174–179.
Wang, Q., Yang, S., Liu, M., Cao, Z., Ma, Q., 2014. An eye-tracking study of website complexity from cognitive load perspective. Decis. Support Syst. 62, 1–10. https://doi.org/10.1016/j.dss.2014.02.007.
Wohltjen, S., Wheatley, T., 2024. Interpersonal eye-tracking reveals the dynamics of interacting minds. Front. Hum. Neurosci. 18. https://doi.org/10.3389/fnhum.2024.1356680.
Xu, F., Correia, A.P., 2023. Adopting distributed pair programming as an effective team learning activity: a systematic review. J. Comput. High. Educ. 0123456789. https://doi.org/10.1007/s12528-023-09356-3.
Yamaguchi, T., Matsuzawa, Y., Niimi, A., Oba, M., 2022. Cycles in state transition as trial-and-errors in solving programming exercises. In: Proceedings of the IFIP World Conference on Computers in Education, pp. 542–553.
Yenigalla, L., Sinha, V., Sharif, B., Crosby, M., 2016. How novices read source code in introductory courses on programming: an eye-tracking experiment. In: Schmorrow, D.D., Fidopiastis, C.M. (Eds.), Foundations of Augmented Cognition: Neuroergonomics and Operational Neuroscience. Springer International Publishing, pp. 120–131.