# Group Programming in COVID-19 Time: The Experience of the Students of the ESI-CR of the UCLM

Ana Isabel Molina Díaz
*Dpto. Tecnologías y Sistemas de Información*
*Universidad de Castilla-La Mancha*
*Ciudad Real, España*
anaisabel.molina@uclm.es

Carmen Lacave Rodero
*Dpto. Tecnologías y Sistemas de Información*
*Universidad de Castilla-La Mancha*
*Ciudad Real, España*
carmen.lacave@uclm.es

***Abstract*—Collaborative learning activities have become a common practice in current university studies due to the implantation of the EHEA. But the COVID-19 pandemic has led to a radical and abrupt change in the teaching-learning model used in most universities, based on a face-to-face model. A rapid and unexpected adaptation to a new model of non-presential teaching has been required, which has been able to be implemented, even in an improvised manner, thanks to the effort of teachers and students. Given this new scenario, our interest is mainly focused on discovering to what extent our students of the Computer Engineering Degree have approached the group programming tasks, which they must perform in a large number of subjects. For this purpose, we have carried out an experience aimed at finding the strategies and software tools they have used to address these tasks. The results of the study indicate that students have adopted a programming model based on work division or distributed peer programming, and very few have chosen to make use of synchronous distributed collaboration tools.**

***Keywords*— Group programming, Peer Programming, *groupware*, COVID-19**

## I. INTRODUCTION

The implantation of the European Higher Education Area (EHEA) has led to group activities becoming a common practice for university students [1, 2]. Specifically, the Degree in Computer Engineering (GII) at the University of Castilla-La Mancha (UCLM) promotes the development of group programming projects of small and medium size in most subjects [3]. Working groups are usually formed by two students who must cooperate to develop some program or practical project. The most frequent ways to approach such joint activities are the distribution of programming tasks in different parts of the program/project (different files, modules or functions), the use of shared repositories (*Git*, *GitHub*, *Google Drive*, *OneDrive*, ...) or the application of *Peer Programming* (PP) techniques [4], mainly in the context of face-to-face laboratory classes.

*Pair Programming* is the term used to describe the process followed by two programmers working in the same computer, performing a particular programming task or the design of an algorithm. In this scenario two roles are defined: the *driver*, who controls the programming activity and is responsible for writing the source code; and the *observer*, who gives indications to his/her partner about the development being carried out, the existence of possible syntax errors, etc. Both roles can be exchanged, alternating the control each team member during the programming activity. Several studies have proved that the use of the PP technique improves the process of solving programming problems, the productivity of the team, and the quality of the programs generated [5, 6].

Pair Programming also requires working face-to-face in the same location. When it is carried out in a distributed environment, it is called *Distributed Pair Programming* (DPP) [7]. In this case, both team members collaborate synchronously on the same programming task, but they are geographically distant, so they must use specific collaboration support tools (*groupware*) to develop their work [8]. When the number of programmers is not limited to two, the technique is known as *Collaborative Programming* (CP). To ensure the efficiency of this process, the tools used must incorporate support mechanisms to the group activity (coordination, access to shared information, *awareness* in the case of working synchronously, ...) [9, 10, 11].

In the second half of March 2020, the confinement due to the COVID-19 pandemic forced a shift from a face-to-face to an *online* education model from just one day to the next [12, 13]. In the case of the universities, each one provided different tools for teachers and students to address this non-face-to-face modality, facilitating the adaptation of methodologies, planning and evaluation [14].

Within this process of improvised and rapid adaptation, the UCLM decided to maintain the usual platform of *online* communication with students (*Moodle*[1]) and the institutional shared information repository (*MS OneDrive*[2]), as well as to provide the university community with video conferencing applications (*MS Teams*[3]), and video creation and playback (*MS Stream*[4]). The combination of these tools allowed to solve the problems of teacher-student communication in a more or less satisfactory way. However, as teachers of the GII, and in view that *online* teaching has come to stay (to a greater or lesser extent), our interest is focused on knowing how the *student-student* communication was approached in the context of group programming tasks, which our students have had to perform in most of the subjects. It is clear that, in the context of the imposed confinement, students at all universities had to make use of new strategies and work tools to move from a traditional PP-based model to a DPP or CP approach.

Therefore, this article describes the research experience carried out at the Escuela Superior de Informática de Ciudad Real (ESI-CR) of the UCLM, which aims to know the mechanisms, tools and difficulties of the students of the GII,

[1] https://moodle.org/?lang=es
[2] https://www.microsoft.com/es-es/microsoft-365/onedrive/online-cloud-storage
[3] https://www.microsoft.com/es-es/microsoft-365/microsoft-teams/group-chat-software
[4] https://www.microsoft.com/es-es/microsoft-365/microsoft-stream

as well as their subjective perception, to address the group programming tasks they performed during the state of alarm, in the period between March and May 2020. Section 2 describes the details of the experience carried out (questionnaire designed, results and discussion); and section 3 comments on the conclusions drawn from this work and the work to be undertaken in the future.

## II. EXPERIENCE

In an effort to find out how the students of the GII of the ESI-CR carried out their group programming tasks during the decreed state of alarm between March and May in Spain, an experience was carried out with the voluntary and anonymous participation of a total of 112 students (14 in the first year, 49 in the second year, 35 in the third year and 14 in the fourth year). The experience took place during the first half of June and consisted of collecting information of interest by means of a questionnaire, displayed through the *MS Forms*[5] tool, and described below.

### A. Questionnaire

Given the objective of this research, a questionnaire was designed to find out the following aspects:

- *Need for group programming activities in a distributed way*. This item asked whether they had to perform group programming activities during the confinement period.
- *Size of the programming groups.* Although in most cases the groups consisted of two members, it was asked whether the groups were made up of two, three or more members.
- *How they have approached group programming tasks.* This item inquired about the solution adopted to perform group programming tasks when the members of the work team could not meet face-to-face. Several answer options were provided (Table I), and several of them could be selected. In the case of choosing one of the last four options, an additional question was enabled to indicate which tool/s they had used.

  Note that option (d) is the one that best aligns with the PP approach, but in a distributed (*online*) format: the videoconferencing tool allows sharing the development environment (IDE [6]) (for instance, *Eclipse* [7]) so that the members of the couple can alternatively take turns to adopt the roles of *driver* and *observer*. On the other hand, option (e) refers to the use of a *groupware* programming environment, which would allow the application of a CP approach.
- *Subjective perception about different strategies for group programming.* This aspect was integrated by three items in which three possible strategies for group programming were presented (Table II). Each of the possible response options was ranked by means of a 5-level Likert scale, which allowed to indicate the degree of agreement (value closest to 5) or disagreement (value closest to 1) with each one.

TABLE I. SOLUTIONS ADOPTED FOR GROUP PROGRAMMING DURING THE CONFINEMENT PERIOD

| Answer item | Statement |
|---|---|
| (a) | I have chosen not to do group practice. I have chosen or changed (if the subject allowed it) to the modality of individual work |
| (b) | We have distributed the practical work of different subjects so that each member of the group can work individually on each one of them, and not have the need to work together on the same project/program |
| (c) | We have used version control systems (e.g., *Git, GitHub,* ...) to work on the same programming project, but asynchronously (not both at the same time on the same project/file) |
| (d) | We have made use of video conferencing tools (e.g. *MS Teams* or similar), sharing the screen or the IDE |
| (e) | We have made use of a synchronous collaborative software development environment |
| (f) | We have made combined use of some of the above options |

TABLE II. DIFFERENT GROUP PROGRAMMING STRATEGIES OR SCENARIOS

| Answer item | Statement |
|---|---|
| COOP | I believe that it is better to *divide the work* when it is necessary to program in group in the same practical project, i.e., that each group member works in an independent way in a certain component (ex. package, file, class, ...) and, then, the contribution of each one is integrated to the final result |
| ASYNC_COLAB | I think that group programming is best done in an *asynchronous* way (each member of the group working on the same code/project, but at different times), taking turns not to "step on" the work |
| SYNC_COLAB | I think that group programming is best done in a *synchronous* way (both partners working at the same time on the same code), using additional channels of *chat*, video or audio to get organized and make decisions together |

The first scenario proposes the division of work and the progress in programming autonomously, in different sections of the final program or project. In other words, a *cooperation* strategy was presented, in which it is possible to work in a *synchronous* or *asynchronous* way, but in different parts of the code. The second one suggests working on the same code but in different moments of time, that is, it describes a *collaboration* scenario, based on the assignment of shifts and, therefore, *asynchronous*. The last scenario corresponds to a purely *collaborative* and *synchronous* group work.

- *Need for tools to support distributed and synchronous group programming.* This section asked about the need for tools to support collaborative synchronous work, which matches the last scenario described in the previous question.
- *Features and functionalities needed to support synchronous distributed programming activities.* Based on a hypothetical scenario of synchronous distributed programming, a series of features and functionalities that could be considered desirable, and

[5] https://forms.office.com
[6] IDE-*Integrated Development Environment.*
[7] https://www.eclipse.org/

even necessary, for effective and efficient group programming are presented.

Participants were asked to rate the usefulness/need for each of these features or functions on a scale of 1 to 5, with the lower end of the scale (1) representing that it would not be necessary or useful at all, and the upper end (5) indicating that it would be very necessary or very useful. Table III shows the list of features, which includes the main communication tool (text-chat, audio and video), coordination and access control to the shared workspace (blocking of code sections, version control) mechanisms, as well as aspects related to *awareness* (connected users and visual highlighting of access to the shared area) [10]. *Awareness* [15, 16] is the set of visualization techniques that are incorporated into the user interface of collaborative applications to provide information about group activity, that is, visual information about the people the user is working with, the activities they are carrying out and about which part of the shared artefact they are working with.

Finally, an item in which students could indicate any feature or functionality not listed that they considered necessary or useful was included.

TABLE III. USEFUL FEATURES AND FUNCTIONALITIES IN A SYNCHRONOUS DISTRIBUTED COLLABORATIVE PROGRAMMING SCENARIO

| Answer item | Statement "The application should..." |
|---|---|
| IDE | ...be an evolution of a known IDE (e.g. *Eclipse, Netbeans, ...*) |
| CONNECT_USERS | ... show the users connected (identified by their name, avatar, availability status, ...) |
| CHAT | ... include a synchronous communication tool (*chat*) |
| AUDIO | ... have the possibility of communicating by audio with the partner |
| VIDEO | ... have a video channel that would allow to make a videoconference with the partner at the same time that it is being programmed |
| AWARENESS | ...show or visually highlight where the partner is writing/working (using colours, icons, etc.) |
| BLOCKING | ...give the possibility of blocking sections of code when the user is working on the same source code file at the same time |
| VERS_CTROL | ...incorporate a version control system |
| LOG | ...maintain a log or record of each group member's contributions to the final project |

## B. Results

The preprocessing of the data provided by the participants in the questionnaire reduced the sample size to $N$=111. The subsequent analysis of the responses provided very interesting results, which are described below.

Regarding the need to perform group programming tasks during the confinement, 98 of the 111 students (88% of the total) answered positively. Most of these students belonged to 2nd and 3rd year (Figure 1).

As for the size of the groups in which they participated, 52 of the 98 students indicated that they programmed in pairs, 16 in groups of three and 13 in groups of more than three. Another 17 indicated that they were part of several groups of different sizes.

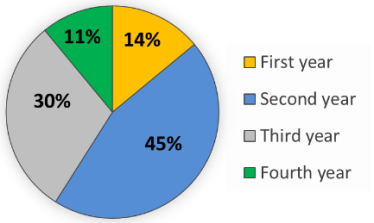


Fig. 1. Distribution by course of the students who expressed the need for group programming during confinement.

The solutions adopted to deal with group programming (Table I) consisted, mostly, in the combination of various strategies, highlighting the use of some videoconferencing system (being *MS Teams* and *Discord* [8] the most cited) together with an asynchronous version control system (*GitHub*[9] was the most outstanding). Figure 2 also shows the percentage of students who chose to use only one tool.

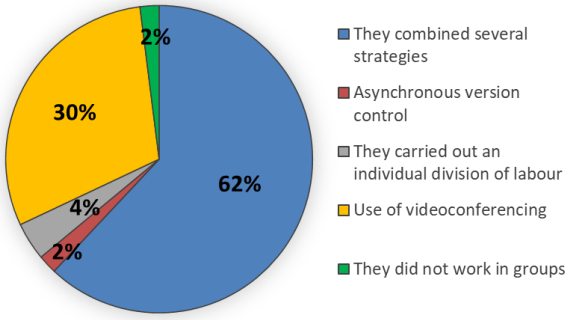


Fig. 2. Percentages illustrating the different solutions adopted for group programming during confinement.

Among those who combined several strategies, five participants from the third and fourth year indicated that they used a synchronous collaborative development environment.

Table IV shows the main descriptive statistics (mean, median and mode) of the answers related to the participants' subjective perception about the convenience of applying the three group programming scenarios described in the previous section (*cooperation*, *asynchronous collaboration* and *synchronous collaboration*) (Table II), the need to have tools to support the last of these three scenarios, in a distributed context (*synchronous distributed collaborative programming*), and, finally, the functionalities and features they considered most necessary or useful in a tool to support this programming strategy (Table III).

As far as the different strategies proposed (Figure 3), the best evaluated was synchronous collaborative programming ($\mu$=4.00), while asynchronous collaboration modality was the worst valued ($\mu$=2.74). Most of the students considered that having tools to support synchronous distributed programming scenarios should be necessary ($\mu$=4.00). The features they

[8] https://discord.com/

[9] https://github.com/

considered most useful for the software supporting this programming strategy (Figure 4) were that the collaborative functionalities should be integrated in a known IDE (e.g. *Eclipse*) ($\mu$=4.05), and that both version control support (μ=4.29) and the recording of individual contributions made by each team member to the final result should be included (μ=4.23). With respect the communication mechanisms that should be incorporated, the best rated was audio (μ=4.21), followed by chat (μ=4.07), with the video signal being the one they considered the least useful (μ=3.26).

TABLE IV. DESCRIPTIVE STATISTICS – PROGRAMMING SCENARIOS AND FEATURES OF SYNCHRONOUS ENVIRONMENTS

| Answer item | Mean ($\mu$) | Std. Dev. | Median | Mode |
|---|---|---|---|---|
| COOP | 3.00 | 1.13 | 3 | 3 |
| ASYNC_COLAB | 2.74 | 1.16 | 3 | 3 |
| SYNC_COLAB | **3.75** | 1.18 | **4** | **4** |
| Need for synchronous collaboration | **4.00** | 1.08 | 4 | 4 |
| IDE | 4.05 | 0.93 | 4 | 5 |
| CONNECT_USERS | 3.88 | 1.17 | 4 | 5 |
| CHAT | 4.07 | 2.06 | 4 | 5 |
| AUDIO | **4.21** | 1.06 | **5** | **5** |
| VIDEO | 3.26 | 1.31 | 3 | 5 |
| AWARENESS | **4.36** | 0.87 | **5** | **5** |
| BLOCKING | 4.19 | 1.04 | 5 | 5 |
| VERS_CTROL | **4.29** | 0.87 | **5** | **5** |
| LOG | **4.23** | 0.93 | **4** | **5** |

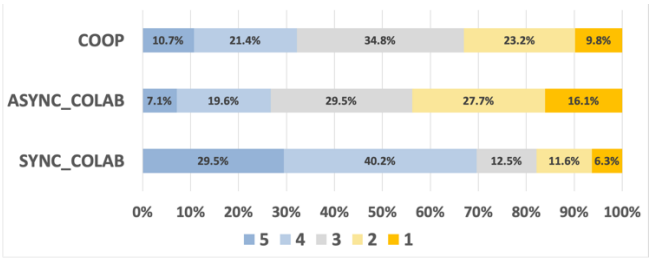


Fig. 3. Assessment of different strategies or group programming scenarios.

The incorporation of *awareness* mechanisms was considered very necessary ($\mu$=4.36), although the possibility of having visible information about the users connected or their availability at any given time ($\mu$=3.88) was considered not very useful.

Finally, we analysed whether the assessment of each of these features depends on some factors, such as the course in which the student was enrolled or the size of the groups in which they had worked. The ANOVA of the data reflected significant differences at 95% (*p-value*=0.003) only in the version control tool when considering the student's course as a factor. The subsequent *post-hoc* revealed that these differences occur between students of the 1st and 2nd courses with respect to those of the 3rd and 4th courses. These results are in line with the data reflected in the section on the solutions adopted, in which only 30 out of 98 students have selected the use of version control tools (*GitHub*, for the majority), among which there are no first-year students and 68% are third year students.

### C. Discussion

The results obtained show that, despite the fact that students positively valued synchronous distributed programming (CP), they have opted, as a first option, for a PP (*driver-observer* roles) programming model, but in an *online* mode (DPP) during the confinement period. Most of them have made use of *MS Teams* or *Discord*, sharing the development IDE (*Eclipse* for *Java*; *MS Visual Studio*[10] for *Visual Basic*, *RStudio*[11] for *R* and *Visual Studio Code*[12] for *C* and *ADA*) and changing turns alternatively to code. In the same way that teachers have opted to transfer the face-to-face magisterial class model to *online* support, in what has been referred as *remote emergency teaching* [14], students have opted for a similar approach. In most cases, they extrapolated the way they work in the practice laboratories to a distributed model. Very few students made use of a distributed and synchronous programming environment, possibly due to a lack of knowledge of the one that would suit their needs. In the very few cases that they did so, the tools used were *Google Colab*[13] for *Python* programming and *MS Visual Studio Live Share*[14] or *Atom*[15] for *C* and *ADA* programming.

Among the desirable features included in a software for group programming, at the same time and in a distributed way, they considered that having an audio channel can be very useful and, possibly, the most agile method to communicate. The video signal is not considered as very necessary, being in many cases rather a source of distraction, while textual communication through a *chat*, that they are very used to, is also well valued.

Version control and the possibility of recovering previous states of the practical projects were highly appreciated by students, for their obvious usefulness [17], although the fact that no first-year students and very few second-year students used them suggests that using this type of tool requires a certain "maturity" not only in the use of technology, but also in how to address group work. Therefore, and considering the advantages that the use of these kinds of tools could offer to GII students [18], it would be necessary to consider for the future introducing students to the use of version control systems in the first or second year.

On the other hand, those who best value version control tools also consider it necessary to record the individual contributions of each team member to the final result. This feature, in order to evaluate and justify the personal involvement in the deliveries, is very useful for both teachers and students.

[10] https://visualstudio.microsoft.com/es/vs/
[11] https://rstudio.com/
[12] https://code.visualstudio.com/
[13] https://colab.research.google.com/notebooks/intro.ipynb
[14] https://visualstudio.microsoft.com/es/services/live-share/
[15] https://atom.io/

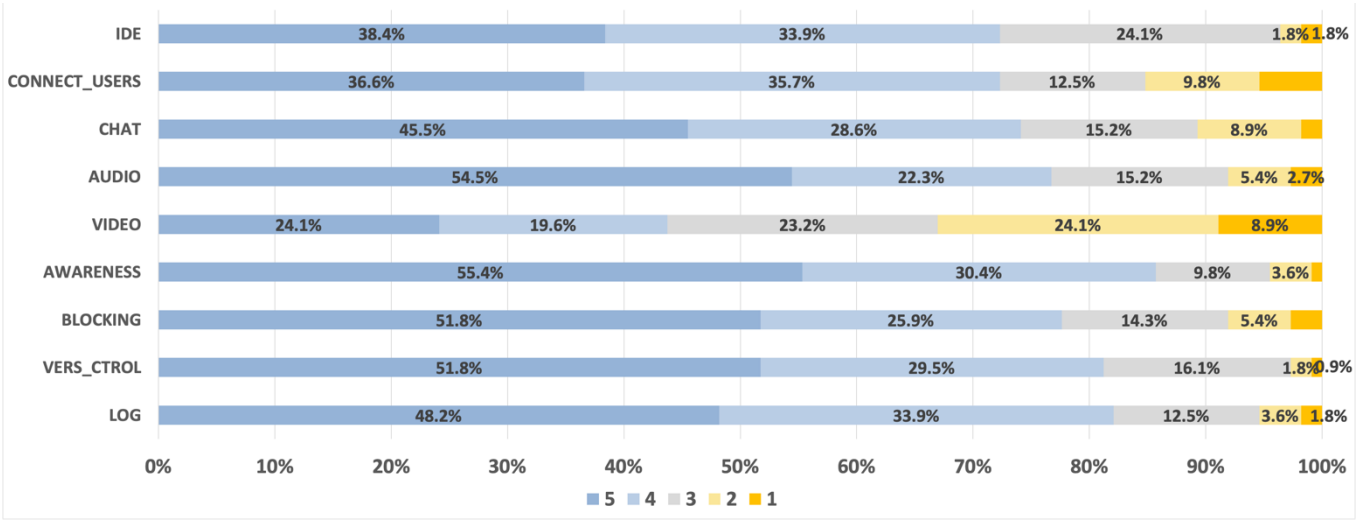


Fig. 4. Assessment of different features or functionalities needed in a synchronous collaborative programming scenario.

In our research group (CHICO[16]) the support of group programming has been, for years, a topic of interest [9, 10, 19]. Among the latest developments, the COLLECE 2.0[17] system stands out. This is a synchronous collaborative programming environment that incorporates many of the features that have been most appreciated by the students in this study (it is a plug-in integrated in *Eclipse*, which incorporates *awareness* mechanisms, version control, blocking of code regions, ...). Although several pilot experiences of the use of this system have been carried out with students of the ESI-CR [20], this software has not yet been implanted as a tool of habitual use in class, so we are considering to use it in several programming subjects during the course 2020-2021.

## III. Conclussions

In this article we have described an experience conducted with more than a hundred students of the Computer Engineering Degree of the ESI-CR of the UCLM, whose objective was to know how they had approached the practical tasks of group programming during the state of alarm decreed in the second quarter of the 2019-2020 academic year.

The results obtained show that, although the students considered interesting the use of synchronous collaborative programming tools, that is, to apply a CP approach, they mostly opted for a DPP model (in which they share the IDE with their colleagues, making use of videoconferencing applications). The division of programming tasks in different parts of the program or project was the second most used option. Possibly the lack of knowledge of support tools for distributed synchronous programming influenced the choice of these strategies.

Our research group has developed several environments that implement the three programming approaches (PP, DPP and CP). Outstanding among these systems is COLLECE 2.0, which we plan to introduce in several programming subjects during the next academic year. This system incorporates many of the features considered most useful by the participants in this study (it is integrated a widely used IDE, such as *Eclipse*, incorporates a version control system, blocking of code regions, a very rich set of *awareness* mechanisms, ...). Even so, some of the features that have also been positively valued by the students could be added to this system, such as the incorporation of an audio channel between the team members. Similarly, we plan to continue studying the tools that support CP through a systematic literature review, which allows us to know the state of current research in this field.

## Acknowledgements

This work has been funded by the Ministry of Economy, Industry and Competitiveness and by the European Regional Development Fund, with reference TIN2015-66731-C2-2-R.

The authors would also like to express their gratitude to the students who volunteered to participate in this experience.

## References

[1] MECD. "La Integración del Sistema Universitario Español en el Espacio Europeo de Educación Superior", Ministerio de Educación, Cultura y Deporte, 2003.

[2] R. Olanda, R. Sebastian, J. I. Panach. Aprendizaje colaborativo basado en tecnologías multimedia. *Proceedings of the XX Jornadas de Enseñanza Universitaria de la Informática (JENUI 2014)*, 2014.

[3] UCLM. "Memoria para la solicitud de verificación de títulos oficiales. Propuesta de título de Graduado en Ingeniería Informática", 2010.

[4] L. Williams, R. R. Kessler. *Pair programming illuminated*. Addison-Wesley Professional, 2003.

[5] S. Faja. Evaluating effectiveness of pair programming as a teaching tool in programming courses. *Information Systems Education Journal*, 12(6), 36, 2014.

[6] L. A. Williams, R. R. Kessler. All I really need to know about pair programming I learned in kindergarten. *Communications of the ACM*, 43(5), 108-114, 2000.

[7] P. Baheti, E. Gehringer, D. Stotts. Exploring the efficacy of distributed pair programming. *In Proceedings of the Conference on Extreme Programming and Agile Methods* (pp. 208-220). Springer, Berlin, Heidelberg. 2002.

[8] B. J. da Silva Estácio, R. Prikladnicki. Distributed pair programming: A systematic literature review. *Information and Software Technology*, 63, 1-10, 2015.

[9] C. Bravo, R. Duque, J. Gallardo. A groupware system to support collaborative programming: Design and experiences. *Journal of Systems and Software*, 86(7), 1759-1771, 2013.

[10] A. I. Molina, J. Gallardo, M. A. Redondo, C. Bravo. Assessing the awareness mechanisms of a collaborative programming support system. *Dyna*, 82(193), 212-222, 2015.

[11] F. Jurado, A. I. Molina, M. A. Redondo, M. Ortega. Cole-programming: Shaping collaborative learning support in eclipse. *IEEE Revista Iberoamericana de Tecnologías del Aprendizaje*, 8(4), 153-162, 2013.

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

[17] http://blog.uclm.es/grupochico/proyecto-iapro/collece-2-0/

[12] H. Fardoun, M. Yousef, C. González-González, C.A. Collazos. Estudio exploratorio en iberoamérica sobre procesos de enseñanza-aprendizaje y propuesta de evaluación en tiempos de pandemia. *Education in the Knowledge Society* (EKS), 21, 9, 2020.

[13] A. Skulmowski, G. D. Rey. COVID-19 as an accelerator for digitalization at a German university: Establishing hybrid campuses in times of crisis. *Human Behavior and Emerging Technologies*, 2020.

[14] AENUI. "Declaración de AENUI sobre retos educativos para el curso 2020-2021". http://www.aenui.net/ (Junio, 2020).

[15] P. Dourish, V. Bellotti. Awareness and coordination in shared workspaces. *In Proceedings of the 1992 ACM conference on Computer-supported cooperative work* (pp. 107-114), 1992.

[16] C. A. Collazos, F. L. Gutiérrez, J. Gallardo, M. Ortega, H. M. Fardoun, A. I. Molina. Descriptive theory of awareness for groupware development. *Journal of Ambient Intelligence and Humanized Computing*, 10(12), 4789-4818, 2019.

[17] K. M. Ying, K. E. Boyer. Understanding Students' Needs for Better Collaborative Coding Tools. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (pp. 1-8), April, 2020.

[18] Y. Lu, X. Mao, T. Wang, G. Yin, Z. Li. Improving students' programming quality with the continuous inspection process: a social coding perspective. *Frontiers of Computer Science*, 14(5), 1-18, 2020.

[19] M. Ortega, M. A. Redondo, C. Bravo, A. I. Molina, C. Lacave, Y. Arroyo,... and D. Fuentes. CHICO 2019 (Computer–Human Interaction and Collaboration), UCLM. *IE Comunicaciones*, 30(30), 2019.

[20] C. Lacave, M. A. García, A. I. Molina, S. Sánchez, M. A. Redondo, M. Ortega. COLLECE-2.0: A real-time collaborative programming system on Eclipse. *In Proceedings of the XXI International Symposium on Computers in Education* (SIIE 2019) (pp. 1-6). IEEE, 2019.