electronics

MDPI

Article

The Impact of COVID-19 in Collaborative Programming. Understanding the Needs of Undergraduate Computer Science Students

Carmen Lacave * and Ana Isabel Molina

CHICO Research Group, Universidad de Castilla-La Mancha, 13071 Ciudad Real, Spain; anaisabel.molina@uclm.es

* Correspondence: carmen.lacave@uclm.es

Abstract: Collaborative learning activities have become a common practice in current university studies due to the implantation of the EHEA. However, the COVID-19 pandemic has led to a radical and abrupt change in the teaching–learning model used in most universities, and in the way students' group work is carried out. Given this new situation, our interest is focused on discovering how computer science students have approached group programming tasks. For this purpose, we have designed a cross-sectional pilot study to explore, from both social and technological points of view, how students carried out their group programming activities during the shutdown of universities, how they are doing them now, when social distance must be maintained, and what they have missed in both situations. The results of the study indicate that during the imposed confinement, the students adopted a programming model based on work division or distributed peer programming, and very few made use of synchronous distributed collaboration tools. After the lockdown, the students mostly opted for a model based on collaborative programming and there was an increased use of synchronous distributed collaboration tools. The specific communication, synchronization, and coordination functionalities they considered most useful or necessary were also analyzed. Among the desirable features included in a software for synchronous distributed programming, the students 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, to which they are very accustomed, is also well valued. In addition, version control and the possibility of recovering previous states of the practical projects were highly appreciated by the students, and they considered it necessary to record the individual contributions of each member of the team to the result.

Keywords: group programming; distributed collaborative programming; students' needs; COVID-19

check for updates

Citation: Lacave, C.; Molina, A.I. The Impact of COVID-19 in Collaborative Programming. Understanding the Needs of Undergraduate Computer Science Students. *Electronics* **2021**, *10*, 1728. https://doi.org/10.3390/electronics10141728

Academic Editor: Dimitris Apostolou

Received: 21 June 2021
Accepted: 12 July 2021
Published: 19 July 2021

1. Introduction

The ability to work in a team is undeniably a basic skill for successful employment in STEM [1] and in particular in computer science (CS) [2]. This is more evident in the field of software development, as it is considered to be a significantly creative and collaborative process [3], and it has been proved that programming in groups improves the process of solving software projects, the quality of the programs generated [4,5], and the programmers' confidence [6]. Therefore, universities should prepare their students effectively for the job market by including the opportunity for teamwork in their curricula [7]. As a result, most undergraduate CS programs related to software development are designed to include the completion of various programming projects of small and medium size, which has been found to have a positive effect on students: they enjoy the social interaction resulting from collaborative activities, and improve their engagement, retention, and performance [8].

Thus, in the first year, when the basics of programming are introduced, the concept of software development is considered as a programming problem, although also involving planning the solution, sharing ideas, organizing files, and editing code. The most frequent ways to approach such joint activities are the distribution of programming tasks in different parts of the program (different files, modules or functions) or the application of pair programming (PP) techniques [9], which constitutes a pedagogical strategy widely used in introductory programming courses [10].

Pair Programming is the term used to describe the process followed by two programmers working on the same computer, performing a specific programming task, or designing an algorithm. In this case, two roles are defined: the driver, who controls the programming activity and writes the source code; and the observer, who gives indications to their partner about the development being carried out, the existence of possible syntax errors, etc. Both roles can be exchanged, alternating the control each member of the team during the programming activity. 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) [11]. In this case, both members of the team collaborate *synchronously* on the same programming task, although they are geographically distant. Thus, they need to use specific collaboration support tools (groupware) to develop their work, mainly screen-sharing applications, or real-time editing software, which provides instantaneous synchronization of edits of all users as they write their code [6]. *Asynchronous* collaboration is generally supported using shared or control version repositories, as well as the incorporation of notification mechanisms [6].

When the number of programmers is not limited to two, the technique is known as *Collaborative Programming* (CP). Although pair programming is the most widely studied form of collaborative activity, increased individual performance appears to result more reliable than other forms [8]. To ensure the efficiency of this process, the tools used must incorporate support mechanisms to the group activity (coordination, access to shared information and workspace, *awareness* in the case of working synchronously, etc.) [12]. *Awareness* [13,14] 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 with whom the user is working, the activities they are carrying out and about which part of the shared artefact they are working with.

In March 2020, the measures to stop the spread of the COVID-19 virus resulted, among others, in the closure of universities, forcing an overnight switch from face-to-face to an online education model [15]. Each university provided teachers and students with different tools to address this non-face-to-face modality, facilitating the adaptation of methodologies, planning and evaluation [16]. In general, the technological ecosystems of the universities made it possible to manage an unprecedented situation [16]. Despite all the challenges posed by the coronavirus pandemic, some universities reopened as safe places by enforcing control measures, such as social distancing, the use of face masks quarantine, etc. preventing group work as it was conceived before the pandemic.

It is still too early to know the real impact that the coronavirus pandemic will have on the education system, especially within the CS community [17] in which collaborative work plays a key role. Recent studies have focused on improving student collaboration in the context of group work and pair programming [18] although students' perceptions and needs about teamwork have not been sufficiently studied [7], especially those arising from the need to use new strategies and work tools to move from a traditional PP-based model to a DPP or CP approach. On the other hand, current trends in collaborative work propose that the challenge be not only technical but also social [19].

Then, we try to shed some light on these issues, focusing on understanding how students have approached their collaborative programming tasks and the emerging needs in these new situations. To this end, we have designed a cross-sectional pilot study at a Spanish public university to explore, both socially and technologically, how the students carried out their group programming activities during the closure of the university, how they do them now, when they must maintain social distance, and what they have missed in both situations. Thus, from the social point of view we have studied the role played by the students' scaffolding, the size of the working groups, and the method used to address collaboration within the group. From the technical point of view, we have studied the groupware tools used and the students' subjective perception about them [8], focusing on their needs since it seems that they are not met by current tools [6]. In addition, we also wanted to investigate whether the distributed collaborative programming model imposed by COVID-19 is here to stay.

The paper is organized as follows: research questions are described in Section 2; the non-experimental design is presented in Section 3; Section 4 describes the results obtained, and Section 5 addresses the discussion, including the main limitations. Finally, Section 6 presents a set of conclusions and further works.

2. Research Questions

The aim of the research is to understand the students' needs imposed by the sanitary restrictions due to the coronavirus pandemic to solve their group programming tasks in a distributed way. To that end, we intend not only to identify the mechanisms and tools used during and after the imposed lockdown but also the difficulties and the subjective perceptions of CS students to address their programming tasks in a distributed way.

Therefore, we have defined the following research questions:

- RQ1: Do the students need to perform their *group programming activities in a distributed way*?
- RQ2: What was the *size of the existing programming group*?
- RQ3: *How have they approached group programming tasks*?
- RQ4: Which was the *students' subjective perception* of the different strategies adopted for group programming?
- RQ5: Do students *require tools that support distributed and synchronous group* programming activities?
- RQ6: Which *features, and functionalities should be useful for students* to support synchronous distributed programming activities?
- RQ7: Are there significant *differences in the students' needs or perceptions* depending on the *enrollment year* or *the size of their programming groups*?

3. Method

In this section, we describe the design of the cross-sectional pilot study aimed at finding out how the COVID-19 pandemic has led to the emergence of new needs in CS students when performing group programming tasks and the solutions adopted. The study, summarized in Figure 1 was conducted based on two experiences carried out at two different moments: the first one was performed in June 2020, immediately after the confinement decreed by the state of emergency in Spain (from March to May 2020), coinciding with the end of the second semester of the academic year 2019/2020; and the second one was conducted in January 2021, at the end of the first semester of the academic year 2020/2021.

In both cases, the information of interest provided by the participants was collected by the same measure instrument: an ad hoc questionnaire. The only difference between the two experiences was the reference period to answer the questions: in the first one, the considered context was the lockdown; in the second, there is no longer imposed confinement, but there were sanitary measures, such as the use of masks, social distancing, and self-confinement in case of positive PCR or close contact with a patient who had tested positive.

Afterwards, the data obtained were analyzed by different statistical methods, making use of IBM Statistics version 25. Since the research was designed as a cross-sectional study rather than as longitudinal, some of the participants may have answered both questionnaires and others only one, although they cannot be differentiated from each other because all data have been obtained anonymously.

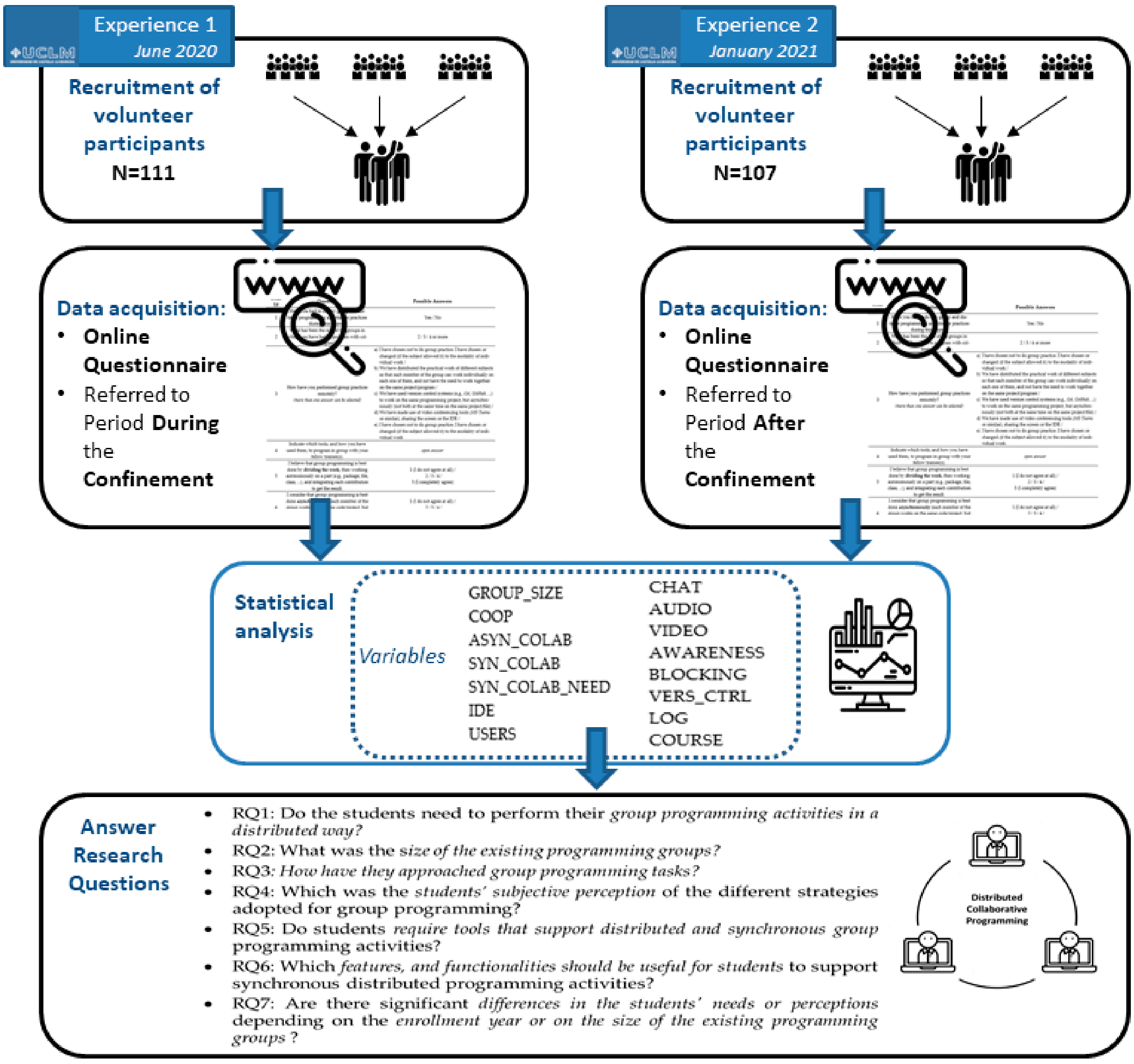
Experience 1
June 2020
UCLM
Recruitment of volunteer participants
N=111
Experience 2
January 2021
UCLM
Recruitment of volunteer participants
N=107
Data acquisition:
• Online Questionnaire
• Referred to Period During the Confinement
Data acquisition:
• Online Questionnaire
• Referred to Period After the Confinement
WWW
Statistical analysis
Variables
GROUP_SIZE
COOP
ASYN_COLAB
SYN_COLAB
SYN_COLAB_NEED
IDE
USERS
CHAT
AUDIO
VIDEO
AWARENESS
BLOCKING
VERS_CTRL
LOG
COURSE
Answer Research Questions
• RQ1: Do the students need to perform their group programming activities in a distributed way?
• RQ2: What was the size of the existing programming groups?
• RQ3: How have they approached group programming tasks?
• RQ4: Which was the students' subjective perception of the different strategies adopted for group programming?
• RQ5: Do students require tools that support distributed and synchronous group programming activities?
• RQ6: Which features, and functionalities should be useful for students to support synchronous distributed programming activities?
• RQ7: Are there significant differences in the students' needs or perceptions depending on the enrollment year or on the size of the existing programming groups ?
Distributed Collaborative Programming

Figure 1. Experiment design.

3.1. Participants and Context

The target population was undergraduates of the computer science degree (CSD) at the Ciudad Real campus of the Spanish public University of Castilla-La Mancha (UCLM). Only CSD students were invited because, during home confinement, the authors of the study did not have the possibility to contact other degree students. In each experience, however, we attempted to gather as many participants as possible, thus, almost all students of the degree (some 500 out of the approximately 650 enrolled) were asked to participate on a voluntary basis.

All candidates received a personal message through the usual online communication platform provided by our university (*Moodle* (https://moodle.org/?lang=en (accessed on 18 July 2021))) by which they were informed about the research work and their participation was requested. They were also informed that, if participating, their data would be treated anonymously and would be used exclusively for research purposes.

Moreover, to facilitate the participation of the students who voluntarily decided to take part in the study, the message contained the link to the questionnaire defined to collect the data, which was displayed through the *MS Forms* (https://forms.office.com (accessed on 18 July 2021)) tool. The questionnaire was accessible to the students for one week and they could fill it in without time limit.

The heading of the questionnaire expressly stated that answering the questionnaire implied the participant's consent to use his/her data exclusively for research purposes. Furthermore, the data were collected completely anonymously, and the participants did not have to enter any information that could identify them, such as name, ID number, etc.

The response rate was about the same in both experiences, approximately 22%.

3.2. Measure Instrument Design

The instrument used to compile the student feedback was an ad hoc questionnaire consisting of 19 items classified into 7 dimensions, as shown Table A1 of the Appendix A. The first six dimensions are related to the research questions from RQ1 to RQ6; the last one is related to demographic information about the students, which is useful to answer RQ7 question and could help to discuss some results.

The first two items try to give an answer to the first two research questions RQ1 and RQ2, respectively, that is, whether the students had to perform group programming activities (Item 1) and whether the groups were made up of two, three, or more members (Item 2). In addition, Item 2 is also useful for research question RQ7.

The third research question, RQ3, has been addressed by the definition of two items, one to know how the students approached group programming tasks in terms of the solution adopted to perform them when the members of the work team could not meet face-to-face (Item 3), and an additional question to indicate which tool(s) they had used (Item 4). Regarding Item 3, several answer options were provided to reflect the different possibilities. Note that option (d) is the one that best aligns with the PP approach, but in a distributed (online) format: the videoconferencing tool allows for sharing the development environment (IDE) (for instance, Eclipse) so that the members of the pair 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.

In order to find an answer to the research question RQ4, related to the subjective perception of different strategies for group programming, three items were defined, each one corresponding to three possible strategies for group programming: divide the programming task among the members of the group (Item 5), work asynchronously on the same code (Item 6) or work synchronously making use of several communication tools (Item 7). Each of the possible response options was ranked by means of a 5-level Likert scale, which allowed the students to indicate the degree of agreement (value closest to 5) or disagreement (value closest to 1) with each one.

Research question RQ5 is intended to know the need for tools to support distributed and synchronous group programming, which matches the last case described in the previous question. Then, an item was defined to ask about the need for collaborative synchronous work (Item 8).

Regarding the answers to research question RQ6, about the features and functionalities needed to support synchronous distributed programming activities [20], several items were defined. The starting point was a hypothetical case of synchronous distributed programming, and then, a set of features and functionalities that could be considered desirable, and even necessary, for effective and efficient group programming were presented (Items 9 to 17) [11]. Those items include the main communication tool (text-chat, audio, and video), coordination and access control to the shared workspace (lock of code sections, version control) mechanisms, as well as aspects related to *awareness* (connected users and visual highlight of access to the shared area) [14,21]. The participants were asked to rate the need for each of these features or functions, or their usefulness 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. In addition, an item in which students could indicate any feature or functionality not listed that they considered necessary or useful was included (Item 18).

Finally, trying to answer the last question RQ7, we included one item to get the highest enrollment year (In Spanish universities, students can be enrolled in different courses at the same time.) for the student (Item 19).

3.3. Variables

Table 1 contains the names of the defined variables which are necessary to perform the subsequent analysis. They represent the collected answers for all closed-ended items of the measure instrument. In addition, one more variable, named TIME, with values {*during_conf, after_conf*} has been defined to denote moment to which the answers refer: during or after the imposed confinement.

Table 1. Variables defined for the statistical analysis.

Item	Question	Variable
2	What is the size of the groups . . . ?	GROUP_SIZE
5	. . . group programming is best done by dividing the work . . .	COOP
6	. . . group programming is best done asynchronously . . .	ASYN_COLAB
7	. . . group programming is best done synchronously . . .	SYN_COLAB
8	. . . there is a need for the development of new tools for distributed and synchronous . . . ?	SYN_COLAB_NEED
9	An environment . . . should be an evolution of a known IDE . . .	IDE_EVOLUTION
10	An environment . . . should display connected users	USERS
11	An environment . . . should have a synchronous communication tool (chat)	CHAT
12	An environment . . . should provide the possibility of audio communication with the partner	AUDIO
13	An environment . . . should have a video channel that would allow videoconferencing . . .	VIDEO
14	An environment . . . should always highlight where the colleague is writing/working . . .	AWARENESS
15	An environment . . . should give the possibility to lock code sections . . .	BLOCKING
16	An environment . . . should incorporate a version control system	VERS_CTRL
17	An environment . . . should keep a history or record of the contributions of each member of the group . . .	LOG
19	What is your highest enrollment year?	COURSE

4. Results

The subsequent analysis of the responses yielded very interesting results, which are described in the next subsections.

4.1. Sample Description

The number of students who voluntarily decided to answer the questionnaire related to the confinement (n = 111) was like the number of students who voluntarily completed it after the confinement (n = 107). Table 2 shows the distribution of students according to the highest enrollment course, together with the percentages relative to each sample size.

Table 2. Sample description.

	during_Conf		after_Conf	
Enrollment Year	*n* = 111		*n* = 107	
1st Year	14	12.6%	22	20.6%
2nd Year	49	44.2%	37	34.6%
3rd Year	34	30.6%	30	28.0%
4th Year	14	12.6%	18	16.8%

4.2. RQ1: Need for Group Programming Activities in a Distributed Way

As Table 3 notes, during the confinement, 98 out of the 111 students (88%) answered positively to Item 1 about the need to perform group programming activities. During this non-confined year, the number of students who answered positively increased to 102 out of the 107 participants (95%). In both cases, most of them were 2nd and 3rd year students.

Table 3. Description of the students who had to program collaboratively.

	during_Conf		after_Conf	
Enrollment Year	98	88%	102	95%
1st Year	14	14.3%	22	21.6%
2nd Year	44	44.9%	37	36.3%
3rd Year	29	30.6%	28	27.4%
4th Year	11	12.6%	15	14.7%

4.3. RQ2: Size of the Existing Programming Groups

During the confinement, 52 out 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. However, after the confinement these values changed significantly because most students indicated that they programmed in groups of three or more members (61 out of 102) and only 8 programmed in pairs. The number of students who took part in more than one group also increased to 33 out of 102. Table 4 summarizes these results and Figure 2 shows the distribution of the size of programming groups according to the enrollment year.

Table 4. Size of the programming group.

	during_Conf		after_Conf	
2 members	**52**	53.1%	8	7.8%
3 members	16	16.3%	**24**	23.5%
More than 3 members	13	13.3%	**37**	36.3%
Several groups of different size	17	17.3%	33	32.4%

4.4. RQ3: How Have They Approached Group Programming TASKS?

The answers provided to Items 3 and 4 indicate that the solutions adopted to deal with group programming (Figure 3) consisted, mostly, in the combination of various strategies, highlighting the use of some videoconferencing system (being *MS Teams* (https://www.microsoft.com/en-gb/microsoft-teams/group-chat-software (accessed on 18 July 2021)) and *Discord* (https://discord.com (accessed on 18 July 2021)) the most cited) together with an asynchronous version control system (*GitHub* (https://github.com (accessed on 18 July 2021)) was particularly outstanding). Figure 3 also shows that a small number of students used only one tool, being the most cited the videoconferencing systems, sharing the IDE, or the screen or the desktop. It is also interesting to note that after confinement, the vast majority used a combination of several tools, and the individual assignment of tasks was not used. Among those who combined several strategies during confinement, 5 participants in 3rd and 4th year indicated that they used a synchronous collaborative development environment, being *CodeTogether* (https://www.codetogether.com (accessed on 18 July 2021)) and *Codeshare* (https://codeshare.io (accessed on 18 July 2021)) the most referenced. However, after confinement that value increased to 26 students: 2 in 1st year, 13 in 2nd, 6 in 3rd, and 5 in 4th.

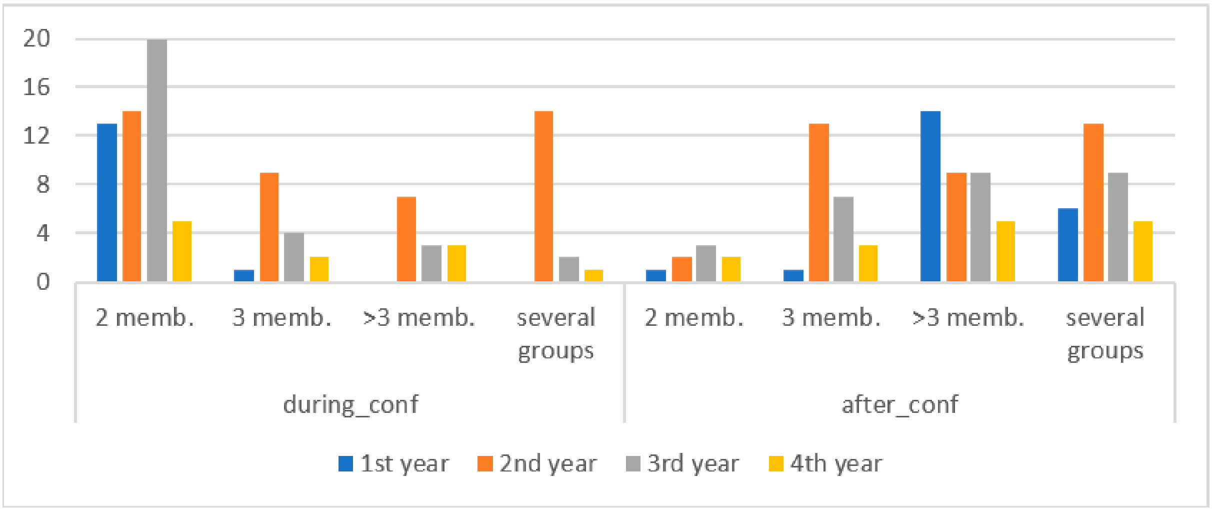

Figure 2. Distribution of the programming group size according to the enrollment year during confinement (**left**) and after confinement (**right**).

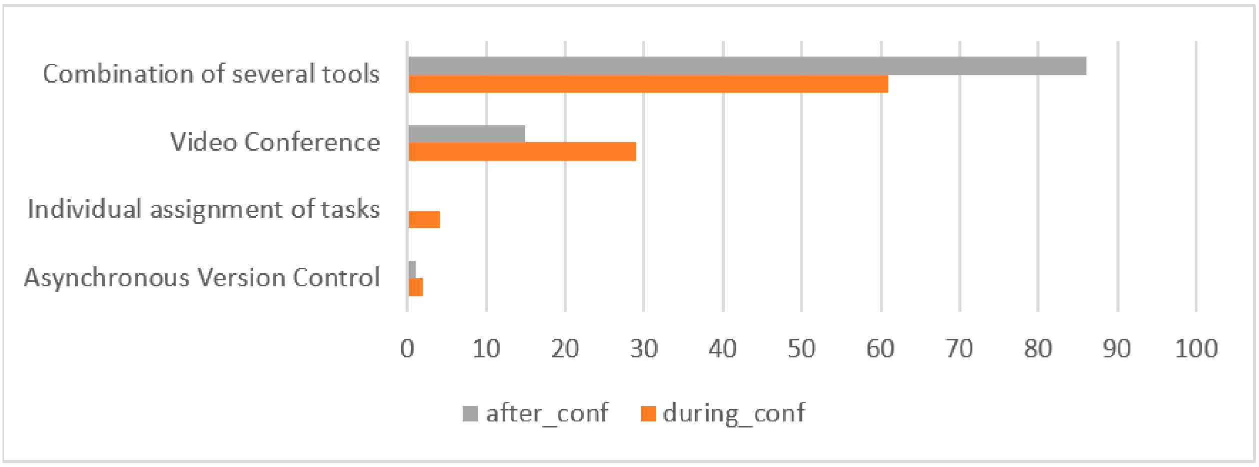

Figure 3. Bar diagram illustrating the different solutions adopted for group programming during confinement (orange) and after confinement (grey).

4.5. *RQ4: Which Was the Students' Subjective Perception of the Different Strategies Adopted for Group Programming?*

Figure 4 summarizes the mean of the answers related to the participants' subjective perception of the convenience of applying the three group programming situations described in the previous section (*cooperation* by work division, *asynchronous collaboration*, and *synchronous collaboration*), according to the enrollment year (Figure 4) and to the size of the programming groups (Figure 5). No significant differences are perceived in the answers given by the students before and after the confinement on which is the best strategy to adopt for group programming. The students would rather prefer to collaborate synchronously (Figure 4), which is more evident in those students whose programming group consists of three members.

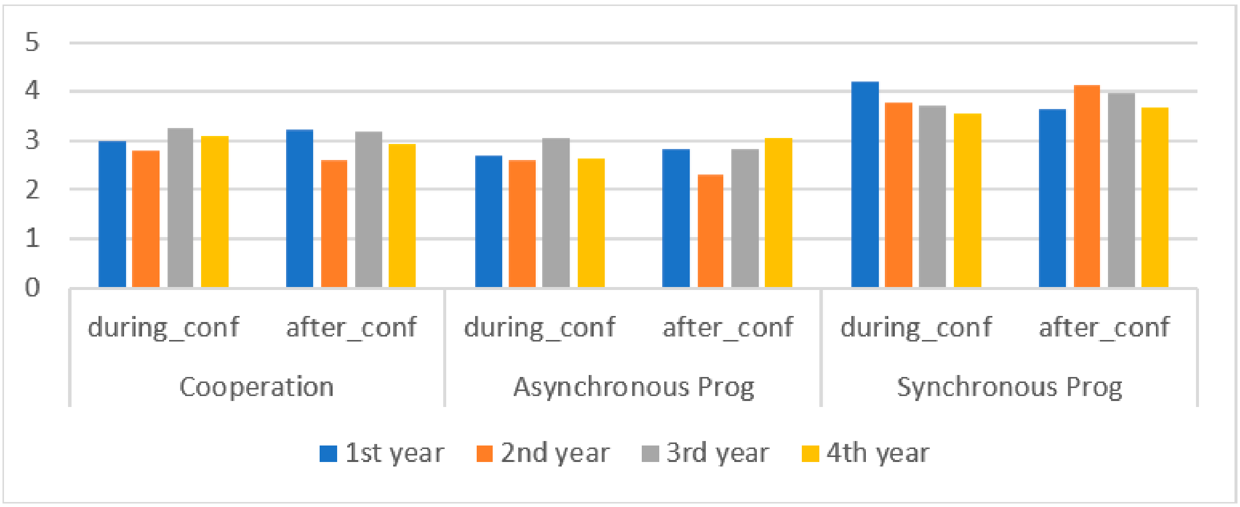

Figure 4. Bar diagram illustrating the subjective perception of the different solutions adopted for group programming during and after the confinement.

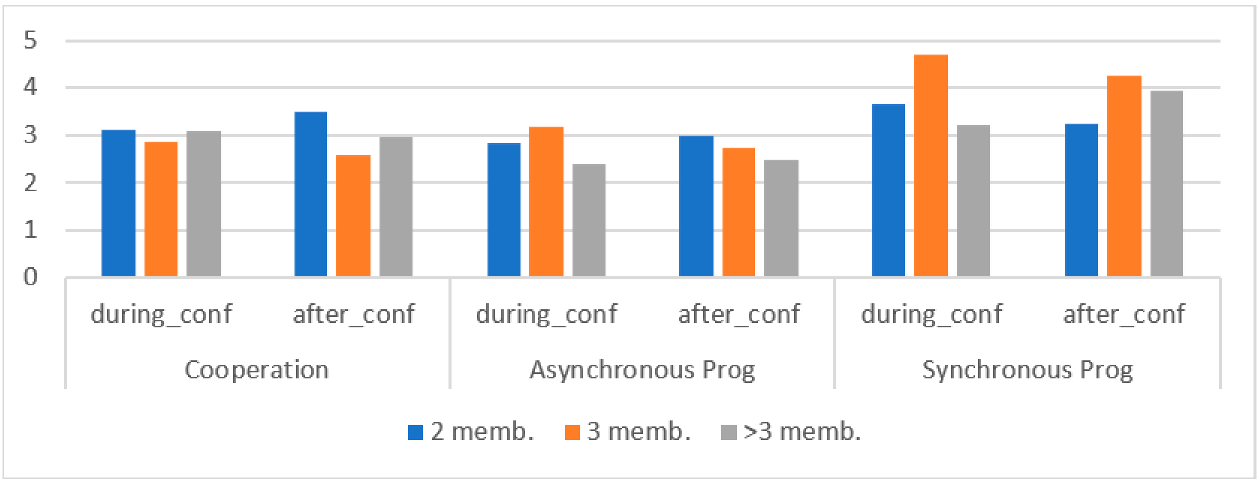

Figure 5. Bar diagram illustrating the subjective perception of the different solutions adopted for group programming regarding the size of the group.

4.6. RQ5: Do Students Require Tools That Support Distributed and Synchronous Group Programming Activities?

In both situations, during and after confinement, the students expressed the necessity of having tools supporting synchronous distributed collaborative programming. This is reflected by the same value of 4.1 for both means of Item 8, as it is shown in Table 5, which also includes the answers for every enrollment year. It can be observed that during confinement, 1st year students are the most in need of synchronous tools and, after the confinement, 2nd and 3rd year students indicate the greatest need.

Table 5. Need for synchronous distributed programming.

	during_Conf		**after_Conf**	
	mean	st. dev.	mean	st. dev.
	4.1	0.925	4.1	1.156
1st year	4.43	0.65	3.82	1.24
2nd year	4.09	0.91	4.19	1.33
3rd year	4.10	1.01	4.25	0.88
4th year	3.73	1.01	4.00	1.13

4.7. RQ6: Which Features and Functionalities Should Be Useful for Students to Support Synchronous Distributed Programming Activities?

There are distinguishing results in both experiences, during and after the confinement. Most of the students considered that having tools to support synchronous distributed programming scenarios should be necessary (μ = 4.00 and μ = 4.10). The features they considered most useful for the software supporting this programming strategy have been the same in both cases: the incorporation of *awareness* mechanisms (μ = 4.36 and μ = 4.58), version control support (μ = 4.29 and μ = 4.31) and the recording of individual contributions made by each member of the team to the result (μ = 4.23). With respect to the communication mechanisms that should be incorporated, the best rated were audio (μ = 4.21 and μ = 4.11), and the possibility of locking code regions (μ = 4.19 and μ = 4.13.), while the video signal was considered the least useful (μ = 3.26 and μ = 2.96). The functionalities and features they considered most necessary or useful in a tool to support this programming strategy are shown in Table 6.

Table 6. Features identified as necessary or useful in a tool supporting synchronous distributed programming.

	during_Conf		**after_Conf**	
	mean	st. dev.	mean	st. dev.
IDE_EVOLUTION	4.05	0.93	3.87	1.00
USERS	3.88	1.17	3.97	1.04
CHAT	4.07	2.06	3.85	1.19
AUDIO	4.21	1.06	4.11	1.17
VIDEO	3.26	1.31	2.96	1.38
AWARENESS	**4.36**	0.87	**4.58**	0.76
BLOCKING	4.19	1.04	4.13	1.05
VERS_CTRL	**4.29**	0.87	**4.31**	0.92
LOG	**4.23**	0.93	**4.23**	1.07

4.8. RQ7: Are There Significant Differences in the Students' Needs Depending on the Enrollment Year or the Size of Their Programming Groups?

The answer to this question involves measuring the variations of the responses given the enrollment course. Since all variables follow a normal distribution, a one-way ANOVA was performed considering the students' year as factor of change. The results reflected no significant differences at 95% although when analyzing data from each experience (during and after the confinement) separately, some interesting findings were observed, which are summarized in Table 7. Hence, during confinement, the ANOVA revealed significant differences at 95% in the means of variables GROUP_SIZE (p = 0.000), USERS (p = 0.011), and VERS_CTRL (p = 0.003).

Table 7. Variables with significant differences in its means according to the enrollment year.

	during_Conf			after_Conf
	GROUP_SIZE	USERS	VERS_CTRL	IDE_EVOLUTION
1st year	2.07	3.86	**3.79**	3.32
2nd year	**3.48**	3.57	4.07	3.97
3rd year	2.55	**4.41**	4.59	3.86
4th year	3.00	4.18	4.73	**4.47**

The subsequent post hoc revealed that, in the case of variable GROUP_SIZE, the differences occurred between 2nd year students with respect to 1st and 3rd year students. This means that the size of the collaborative programming groups of 2nd year students was greater than 1st and 3rd year students. Regarding variable USERS, students of the 3rd year reported greater needs of displaying connected users than students of the 2nd year. As far as variable VERS_CTRL, the differences are between 1st year students with respect to 3rd and 4th year students, because first-year students reported lower requirements regarding the control of the different versions of the program.

After the confinement, only significant differences were detected in variable IDE_EVOLUTION (p = 0.005), between 1st and 4th year students. Consequently, it can be interpreted as that 4th year students have more need than first-year students that the synchronous collaborative programming tool be the evolution of a known IDE.

Regarding the size of the existing programming groups, the one-way ANOVA only detected significant differences at 95% in variable SYNC_COLAB (p = 0.003) during the confinement. This may be because the students who programmed in groups of three (μ = 4.69) reported greater needs for synchronous collaborative tools than those who programmed in groups of two members (μ = 3.67), in groups of more than three members (μ = 3.23), and who formed more than one programming group (μ = 3.76).

5. Discussion

The results obtained also show that, even though the students positively valued synchronous distributed programming (CP), during the confinement period they opted for a PP (driver-observer roles) programming model as a first option, but in an online mode (DPP). 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. However, after the confinement, the increase in the number of students opting for collaborative work and in the number of groups to which they belonged, together with the decrease in the number of pairs, indicate that the students were open-minded about working in groups.

In addition, it seems that the confinement has provided an opportunity to discover different synchronous collaborative tools as well as to become familiar with their use. This is also supported by the fact that the individual assignment of tasks has not been used and by the increase in the percentage of students who have used a synchronous collaborative system (from 5% during the confinement to 25.5% during this school year). Moreover, during the confinement these systems were only used by 3rd and 4th year students, while after the confinement it has been used by students from all years.

Regarding the tools used to collaborate, most of the students made use of *MS Teams* or *Discord*, sharing the development IDE (*Eclipse* for Java; *MS Visual Studio* for Visual Basic, *RStudio* for R and *Visual Studio Code* for C and ADA) and taking turns alternatively to code. In the very few cases that they did so, the tools used were *Google Colab* for Python programming and *MS Visual Studio Live Share* or *Atom* 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, the students considered that 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. However, the possibility that code regions can be locked as they are edited is also well valued.

Version control tools and the possibility of recovering previous states of the practical projects were highly appreciated by the students, for their obvious usefulness [6]. The fact that no first-year students and very few second-year students used them suggests that this type of tool requires a certain "maturity" not only in the use of technology, but also in the way of addressing work in groups because it is a long process [6]. Therefore, considering the advantages that the use of these kinds of tools could offer to CS students [22], they should start using version control systems as early as 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 member of the team to the result. This feature is very useful for both teachers and students to evaluate and justify the personal involvement in the deliveries.

Finally, no significant differences are detected in the students' needs during and after the confinement depending on the enrollment year or the size of their programming groups.

To summarize, the pandemic has highlighted the need to integrate technology into their training models in order to take advantage of existing resources and means, in both in face-to-face and virtual processes [16]. In our research group (CHICO) [23] the support of group programming has been, for years, a topic of interest [24,25]. Among the latest developments, the COLLECE 2.0 system [26] 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, lock of code regions, etc.). 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 members of the team and the possibility of locking code regions. 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. The pilot experiences to evaluate the system [27] are producing very promising results, thus we are considering using it in several programming subjects throughout the academic year 2021–2022.

Limitations

The study presents several threats to validity [28] that might have influenced our results.

- Statistical Conclusion Validity. We have tried to enhance our results by the proper application of the statistical tests. In fact, besides descriptive results, we have checked whether the distributions adjusted to the normal to perform the subsequently parametric analysis. Moreover, we have provided the participants with non-dichotomic variables to avoid the restriction of range and we have tried to identify variability in different sources. However, some uncontrolled extraneous factors, such as personal feelings or circumstances, that could affect the students' responses have been beyond our reach.
- Internal Validity. The results are based on a biased sample formed solely by the students who chose to participate. On the other hand, it is likely that an important part of the participants in the second experience had already done so in the first one, so the repetition of the answers may have influenced the results.
- Construct Validity. Our research questions may not provide complete coverage of variables, e.g., gender [29] or the way the programming group was formed [30,31]. Also, the list of factors used to check alternative explanations was intended to be exhaustive, however additional factors could also be considered (e.g., learning style or general programming experience [31]).
- External Validity. Since the sample is formed exclusively by volunteer students, there is no certainty that the sample is representative of the generality and, consequently,

the results are not generalizable for all college students. Thus, the reproduction of the case study in other context tools remains open as an important line of future work.

6. Conclusions

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 identify the mechanisms and tools used during and after the imposed lockdown and the difficulties and the subjective perceptions of CS students to address their programming tasks in a distributed way.

The experience has revealed the **need for group programming in a distributed way** among CS students, especially in the first three years, although no significant differences were detected in the students' needs depending on the enrollment year or the size of their programming groups.

Concerning the lessons learned from this study, we would like to highlight the good acceptance of collaborative programming tools by CS students. However, the lack of experience with this type of software has led them to move from the traditional model of face-to-face group work to a distributed support. Thus, during confinement, *Distributed Pair Programming* was the preferred **programming paradigm** by the students, although the lockdown gave them the opportunity to discover different synchronous collaborative programming tools, especially for those in the higher grades. So, the return to face-to-face teaching has resulted in an increase of the use of *Collaborative Programming* support tools for upper-level students. Therefore, in the classroom context, it would be advisable to introduce first-year students in the use of *Collaborative Programming* support tools, so that they can use them outside the laboratory (during weekends, holidays), where they should complete their practical work.

Another lesson learned, related to the **communication** needs of group members during confinement, is that video systems have been the most widely used tools for remote work, but students consider that an audio channel might be more effective, as it is more agile and less distracting than video.

Finally, the two most critical aspects considered by students are related to **coordination** support and the recording of individual contributions (**collaboration**) in the shared program production. In terms of coordination, the preferred control mechanism to allow shared code editing was the locking of the program sections to be edited, as opposed to other access policies such as shift assignment, free editing without locking, etc. As for the second aspect, students find it essential to be able to record the individual contributions of each group member, probably because the teacher also evaluates the work done by each one of them. Related to the later aspect, and to the possibility of recovering previous states of the shared artefact, version control systems are highly appreciated by the students from all years, even though they are mainly used by higher levels students. Then, it would be advisable for students to start learning the use of version control systems from the first year.

Our research group has developed several environments that implement the three programming approaches (PP, DPP and CP). Among these systems, it is COLLECE 2.0 that stands out, which incorporates many of the features considered most useful by the participants in this study: a widely used IDE, such as *Eclipse*, is integrated and it incorporates a version control system, lock of code regions, and a very rich set of *awareness* mechanisms.

For future work, we would like to delve more deeply into the role of gender in the context of collaborative programming, from the point of view of the way they communicate as well as the formation of groups and the role they play within it.

Author Contributions: Conceptualization, methodology, and investigation C.L. and A.I.M.; writing, C.L.; writing—review and editing, A.I.M. Both authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Economy, Industry and Competitiveness, and the European Grant Number TIN2015-66731-C2-2-R.

Acknowledgments: We would like to thank to all our students who voluntarily accepted to participate in this experience.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Questionnaire defined to collect data.

Research Question	Item ID	Question	Possible Answers
RQ1	1	Have you had to do any group and distance programming activities or practices during this course?	Yes/No
RQ2	2	What is the size of the groups in which you have had to program with your classmates?	2/3/4 or more
RQ3	3	How have you performed group practices remotely? *(More than one answer can be selected)*	(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 (*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
	4	Indicate which tools, and how you have used them, to program in group with your fellow trainee(s).	*open answer*
RQ4=	5	I believe that group programming is best done by **dividing the work**, then working autonomously on a part (e.g., package, file, class, . . .), and integrating each contribution to get the result.	1 (Strongly disagree)/2/3/4/5 (Strongly agree)
	6	I consider that group programming is best done **asynchronously** (each member of the group works on the same code/project, but at different times), taking turns to avoid overlapping each other's work.	1 (Strongly disagree)/2/3/4/5 (Strongly agree)
	7	I believe that group programming is best done **synchronously** (two colleagues working at the same time on the same code), making use of additional chat, video, or audio channels to organize and make decisions together.	1 (Strongly disagree)/2/3/4/5 (Strongly agree)

Table A1. *Cont.*

Research Question	Item ID	Question	Possible Answers
RQ5	8	Do you consider that there is a need for the development of new tools that allow distributed and synchronous (at the same time) group programming?	1 (Strongly disagree)/2/3/4/5 (Strongly agree)
RQ6	9	An environment that would allow distributed and synchronous (at the same time) programming **should be an evolution of a known IDE** (e.g., *Eclipse*, *Netbeans*, . . .)	1 (would not be at all necessary or useful)/2/3/4/5 (would be very necessary or very useful)
	10	An environment that would allow distributed and synchronous (at the same time) programming **should display connected users (identified by name, avatar, availability status, . . .)**	1 (would not be at all necessary or useful)/2/3/4/5 (would be very necessary or very useful)
	11	An environment that would allow distributed and synchronous (at the same time) programming **should have a synchronous communication tool (chat)**	1 (would not be at all necessary or useful)/2/3/4/5 (would be very necessary or very useful)
	12	An environment that would allow distributed and synchronous (at the same time) programming **should provide the possibility of audio communication with the partner**	1 (would not be at all necessary or useful)/2/3/4/5 (would be very necessary or very useful)
	13	An environment that would allow distributed and synchronous (at the same time) programming **should have a video channel that would allow videoconferencing with the partner while programming.**	1 (would not be at all necessary or useful)/2/3/4/5 (would be very necessary or very useful)
	14	An environment that would allow distributed and synchronous (at the same time) programming **should always visually show or highlight where the colleague is writing/working (by colors, icons, etc.)**	1 (would not be at all necessary or useful)/2/3/4/5 (would be very necessary or very useful)
	15	An environment that would allow distributed and synchronous (at the same time) programming **should give the possibility to lock code sections, when working simultaneously on the same code file**	1 (would not be at all necessary or useful)/2/3/4/5 (would be very necessary or very useful)
	16	An environment that would allow distributed and synchronous (at the same time) programming **should incorporate a version control system**	1 (would not be at all necessary or useful)/2/3/4/5 (would be very necessary or very useful)
	17	An environment that would allow distributed and synchronous (at the same time) programming **should keep a history or record the contribution of each member of the group to the final project.**	1 (would not be at all necessary or useful)/2/3/4/5 (would be very necessary or very useful)

Table A1. *Cont.*

Research Question	Item ID	Question	Possible Answers
	18	In relation to the previous question, do you consider that it would be necessary or useful to incorporate any additional feature or functionality? Indicate which one(s)	*open answer*
RQ7	19	What is the highest year in which you are enrolled?	1st year/2nd year/3rd year/4th year

References

1. McGunagle, D.; Zizka, L. Employability Skills for 21st-Century STEM Students: The Employers' Perspective. *High. Educ. Skill Work-Based Learn.* **2020**, *10*, 591–606. [CrossRef]
2. Exter, M.; Caskurlu, S.; Fernandez, T. Comparing Computing Professionals' Perceptions of Importance of Skills and Knowledge on the Job and Coverage in Undergraduate Experiences. *ACM Trans. Comput. Educ.* **2018**, *18*, 21. [CrossRef]
3. Association for Computing Machinery (ACM) and IEEE Computer Society Joint Task Force on Computing Curricula Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. Available online: https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf (accessed on 18 July 2021).
4. Faja, S. Evaluating Effectiveness of Pair Programming as a Teaching Tool in Programming Courses. *Inf. Syst. Educ. J. ISEDJ* **2014**, *12*, 36–45.
5. Williams, L.A.; Kessler, R.R. All I Need to Know about Pair Programming I Learned in Kindergarten. *Commun. ACM* **1999**, *43*, 108–114. [CrossRef]
6. Ying, K.M.; Boyer, K.E. Understanding Students' Needs for Better Collaborative Coding Tools. In Proceedings of the Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, 25–30 April 2020. [CrossRef]
7. Phillips, H.; Ivins, W.; Prickett, T.; Walters, J.; Strachan, R. Using Contributing Student Pedagogy to Enhance Support for Teamworking in Computer Science Projects. In Proceedings of the Computing Education Practice 2021, New York, NY, USA, 7 January 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 29–32.
8. Luxton-Reilly, A.; Simon; Albluwi, I.; Becker, B.A.; Giannakos, M.; Kumar, A.N.; Ott, L.; Paterson, J.; Scott, M.J.; Sheard, J.; et al. Introductory Programming: A Systematic Literature Review. In Proceedings of the Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, Larnaca, Cyprus, 2 July 2018; pp. 55–106.
9. Williams, L.A.; Kessler, R.R. *Pair Programming Illuminated | Guide Books*; Addison-Wesley: Boston, MA, USA, 2003.
10. Sobral, S.R. Is Pair Programing in Higher Education a Good Strategy? *Int. J. Inf. Educ. Technol.* **2020**, *10*, 7.
11. Da Silva Estácio, B.J.; Prikladnicki, R. Distributed Pair Programming: A Systematic Literature Review. *Inf. Softw. Technol.* **2015**, *63*, 1–10. [CrossRef]
12. Molina, A.I.; Redondo, M.A.; Lacave, C.; Ortega, M. Assessing the Effectiveness of New Devices for Accessing Learning Materials: An Empirical Analysis Based on Eye Tracking and Learner Subjective Perception. *Comput. Hum. Behav.* **2014**, *31*, 475–490. [CrossRef]
13. Dourish, P.; Bellotti, V. Awareness and Coordination in Shared Workspaces. In Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work—CSCW '92, Toronto, Canada, 31 October–4 November 1992; ACM Press: New York, NY, USA, 1992; pp. 107–114.
14. Collazos, C.A.; Gutiérrez, F.L.; Gallardo, J.; Ortega, M.; Fardoun, H.M.; Molina, A.I. Descriptive Theory of Awareness for Groupware Development. *J. Ambient Intell. Humaniz. Comput.* **2019**, *10*, 4789–4818. [CrossRef]
15. UNESCO Education. From Disruption to Recovery 2020. Available online: https://en.unesco.org/covid19/educationresponse (accessed on 18 July 2021).
16. García-Peñalvo, F.J. Digital Transformation in the Universities: Implications of the COVID-19 Pandemic. Available online: https://repositorio.grial.eu/bitstream/grial/2230/1/01.pdf (accessed on 18 July 2021).
17. Crick, T.; Knight, C.; Watermeyer, R.; Goodall, J. The Impact of COVID-19 and "Emergency Remote Teaching" on the UK Computer Science Education Community. In Proceedings of the United Kingdom & Ireland Computing Education Research conference, Glasgow, UK, 3–4 September 2020; pp. 31–37. [CrossRef]
18. Khan, S.; Rabbani, M.R.; Thalassinos, E.I.; Atif, M. Corona Virus Pandemic Paving Ways to Next Generation of Learning and Teaching: Futuristic Cloud Based Educational Model. Available online: https://www.researchgate.net/profile/Eleftherios-Thalassinos/publication/348730084_Corona_Virus_Pandemic_Paving_Ways_to_Next_Generation_of_Learning_and_Teaching_Futuristic_Cloud_Based_Educational_Model/links/600d5b2645851553a06824c6/Corona-Virus-Pandemic-Paving-Ways-to-Next-Generation-of-Learning-and-Teaching-Futuristic-Cloud-Based-Educational-Model.pdf (accessed on 18 July 2021).
19. Chorfi, A.; Hedjazi, D.; Aouag, S.; Boubiche, D. Problem-Based Collaborative Learning Groupware to Improve Computer Programming Skills. *Behav. Inf. Technol.* **2020**, 1–20. [CrossRef]

20. Altebarmakian, M.; Alterman, R.; Yatskar, A.; Harsch, K.; DiLillo, A. The Microgenetic Analysis of Staged Peer Collaboration for Introductory Programming. In Proceedings of the 2016 IEEE Frontiers in Education Conference (FIE), Erie, PA, USA, 12–15 October 2016; pp. 1–8.
21. Molina, A.I.; Gallardo, J.; Redondo, M.Á.; Bravo, C. Assessing the Awareness Mechanisms of a Collaborative Programming Support System. *Dyna* **2015**, *82*, 212–222. [CrossRef]
22. Lu, Y.; Mao, X.; Wang, T.; Yin, G.; Li, Z. Improving Students' Programming Quality with the Continuous Inspection Process: A Social Coding Perspective. *Front. Comput. Sci.* **2020**, *14*, 145205. [CrossRef]
23. Grupo Computer Human Interaction and Collaboration (CHICO). Available online: https://blog.uclm.es/grupochico (accessed on 18 July 2021).
24. Bravo, C.; Duque, R.; Gallardo, J. A Groupware System to Support Collaborative Programming: Design and Experiences. *J. Syst. Softw.* **2013**, *86*, 1759–1771. [CrossRef]
25. Ortega, M. Computer-Human Interaction and Collaboration: Challenges and Prospects. *Electronics* **2021**, *10*, 616. [CrossRef]
26. Lacave, C.; García, M.A.; Molina, A.I.; Sánchez, S.; Redondo, M.A.; Ortega, M. COLLECE-2.0: A Real-Time Collaborative Programming System on Eclipse. In Proceedings of the 2019 International Symposium on Computers in Education (SIIE), Tomar, Portugal, 21–23 November 2019; pp. 1–6.
27. Schez-Sobrino, S.; García, M.Á.; Lacave, C.; Molina, A.I.; Glez-Morcillo, C.; Vallejo, D.; Redondo, M.Á. A Modern Approach to Supporting Program Visualization: From a 2D Notation to 3D Representations Using Augmented Reality. *Multimed. Tools Appl.* **2021**, *80*, 543–574. [CrossRef]
28. Shadish, W.R.; Cook, T.D.; Campbell, D.T. *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*; Houghton Mifflin: Boston, NY, USA, 2001; ISBN 978-0-395-61556-0.
29. Ying, K.M.; Rodríguez, F.J.; Dibble, A.L.; Boyer, K.E. Understanding Women's Remote Collaborative Programming Experiences: The Relationship between Dialogue Features and Reported Perceptions. *Proc. ACM Hum. Comput. Interact.* **2021**, *4*, 1–29. [CrossRef]
30. Revelo Sánchez, O.; Collazos, C.A.; Redondo, M.A. A Strategy Based on Genetic Algorithms for Forming Optimal Collaborative Learning Groups: An Empirical Study. *Electronics* **2021**, *10*, 463. [CrossRef]
31. Sobral, S.R. Pair Programming and the Level of Knowledge in the Formation of Pairs. In *Trends and Applications in Information Systems and Technologies*; Advances in Intelligent Systems and Computing; Rocha, Á., Adeli, H., Dzemyda, G., Moreira, F., Ramalho Correia, A.M., Eds.; Springer International Publishing: Cham, Switzerland, 2021; Volume 1367, pp. 212–221. ISBN 978-3-030-72659-1.