UNCOLAB: A COMPUTATIONAL TOOL FOR COLLABORATIVE LEARNING OF COMPUTER PROGRAMMING

J. Mendez-Lara, J.J. Ramírez-Echeverry, F. Restrepo-Calle

Universidad Nacional de Colombia (COLOMBIA)

Abstract

Computer-Supported Collaborative Learning (CSCL) is an educational proposal that seeks to enable students to achieve their learning goals through technology-mediated group interaction. In the area of computer programming, some studies show increased participation and better results in solving programming problems by students who participate in CSCL learning environments. In university environments, introductory computer programming courses seek to prepare students to solve problems through the use of a programming language; for this purpose, different tools have been created under the CSCL approach, which serve as spaces where students self-regulate their learning, participate voluntarily, discuss ideas and evaluate each other's work. However, according to the software implementations reviewed in the literature, there is still a need to create collaborative tools not only in the early stages of the programming problem solving process, but also during the formative evaluation process of the problem solving through peer collaboration. In this context, the objective of this work is to present the computational tool called UNColab to support programming problem solving through collaboration among students during a peer evaluation stage. A prototype of UNColab was implemented and tested during a collaborative learning activity in a Computer Programming course with 27 students of Systems and Computer Engineering at the *Universidad Nacional de Colombia*. This tool was developed according to a collaborative learning script, which allowed structuring a class dynamic in two activities of the course. The results of this experience suggest that students present a better understanding of the proposed problems. The students expressed their agreement with respect to the usefulness of the tool to support the resolution of programming exercises in a collaborative manner. In addition, they suggested different improvements that could open the possibilities for further studies. Therefore, this work contributes to the research area through the development of a tool that allows supporting the teaching of computer programming in collaborative environments. Also, the validation of the tool in a computer programming course allows a better understanding of the benefits and challenges of using tools for computer-assisted collaborative learning.

Keywords: collaborative learning, computer-supported collaborative learning, computer programming.

1 Introduction

The main objective of introductory computer programming courses is to provide students with the necessary tools, methods and methodologies to develop software solutions based on the use of a programming language. In these courses, the most common way used by teachers to measure the level of success of their students is through the evaluation of a program or programming challenge, which consists of verifying its functionality through different test cases [1]. In the development of these programs, some students face adverse situations, such as lack of understanding of the problem or frustration when performing different unsuccessful attempts, which can affect the student's motivation [2]. However, there are also students who perform well in this type of activity and can achieve the objective of the task without major complications. Consequently, this type of situation in the classroom has allowed different ways of teaching and learning to be proposed; one of them, used in this work, is Computer Supported Collaborative Learning (CSCL). It is an educational proposal that seeks to enable students to achieve their learning objectives through technology-mediated group interaction. Accordingly, CSCL proposes two main ideas: students collaborate with each other to meet their learning objectives; and technologies are adopted to serve as mediating tools in the learning process [3]. Under this perspective, CSCL-based computer programming teaching/learning scenarios allow those students who have difficulties in solving a programming problem to receive help from their peers through software tools.

CSCL tools in computer programming have had good results with respect to student performance in programming courses and the development of some soft skills. For example, in [4], it is concluded a higher educational efficiency and motivation in students when using an application that facilitates communication in the course. Studies such as [5, 6] report an increase in critical thinking skills by

involving students in the process of program evaluation through peer review. Additionally, there are more formal CSCL application studies such as those seen in [7-10], which focus on the use of CSCL scripts to generate support tools in computer programming, which report facilitation in the process of assigning tasks to students, better formation of learning groups and an increase in motivation. However, these studies also suggest further experimentation with CSCL scripts to measure their effectiveness, especially in group assignment and how they affect collaboration among students.

This paper presents the implementation of a prototype of the computational tool called UNColab, which supports the resolution of programming assignments in a collaborative learning activity in a computer programming course. The development of this prototype was based on the framework proposed in [11] for the design of CSCL scripts. These scripts allow establishing a sequence of activities to be performed by a group of students, and they also define how the collaborative work will be structured and the assignment of roles within each activity. The research question posed to solve in this work is: What are the effects on the resolution of computer programming problems from the use of a collaborative computational tool that facilitates formative peer assessment?

This rest of this paper is organized as follows. Section 2 presents the implementation of a CSCL script in the development of the UNColab tool. Section 3 describes the design of the study conducted describing aspects such as the educational intervention, the participants, and the instruments used for data collection. Section 4 presents the results of the study based on the participants' perceptions of the tool developed and the evaluation of the collaborative dynamics. Section 5 discusses these results. Finally, Section 6 presents the conclusions of the work carried out.

2 UNColab

UNColab is a software tool that supports the resolution of computer programming problems through collaboration among students. It is developed so that students who fail to meet the requirements of a programming problem can receive help from one or more classmates. UNColab functions as a complement to the UNCode[1] educational environment, proposed in [12], which supports programming courses by providing academic management functionalities such as student, class, assignment, and submission management. Students use UNCode to submit their source code solutions, since it allows them to have an automatic feedback and assessment on each of the tasks in addition to different add-ons that support the resolution of assignments.

2.1 UNColab Design

UNColab functions were developed from the framework proposed by [13] for the design of CSCL scripts, which establishes four levels of structuring for a computer-assisted collaborative activity. Each level seeks to answer a question related to the collaborative learning activities proposed in a class:

- Collaborative learning flow level. Question that seeks to solve: what is the sequence of steps to organize the collaborative work?
- Activity level. Question to be solved: What activities will the students perform during the collaborative activity?
- Role level. Question that seeks to solve: What role will students play in order to collaborate?
- Resource level. Question to be answered: What resources or tools are available to support the collaboration?

The first three levels must be accompanied by one or more Collaborative Learning Techniques (CLTs), these allow instructing students on group work and offer dynamics that have already been implemented and validated in academic settings [14]. Table 1 presents the list of CLTs that were identified for this study.

Table 1. CTLs for the design of the CSCL script in computer programming for UNColab.

	Collaborative learning flow level	**Activity Level**	**Role Level**
CLTs	Think-pair-share (TPS)	● Introductory activity ● Group discussion	Programmers and novices

The techniques presented in Table 1 provide answers to each of the questions posed above for the

[1] https://uncode.unal.edu.co/

first 3 levels of the CSCL script. At the "Collaborative learning flow" level, TPS allows us to propose collaborative work in 3 phases: think a problem individually, form groups or select a pair and solve collaboratively [15]. At the "Activity" level, it is proposed to use two techniques; the CLT "Introductory activity", which allows establishing the objectives or achievements that the student will reach when solving the task [15]; and "Group discussion", which proposes that the student discusses with a classmate to contribute to the resolution of the task [15]. Finally, at the "Role" level, "Programmers and novices" is proposed; this is a technique based on the strategy proposed in [16], which allows supporting the processes of role assignment and collaboration through the division between students into two categories: those students that manage to solve a programming problem, who are categorized as “programmers”; and those who present difficulties, who are classified as “novices”.

UNColab functionalities have been designed according to the characteristics of the CLTs seen in Table 1. This tool, together with UNCode, establishes the level of resources according to the CSCL script structuring levels. Table 2 presents the list of functionalities that the computational resource proposes to support collaborative learning in computer programming.

Table 2. Functionalities and resources for designing a CSCL script in computer programming.

Functionality	Resource level
Approaching the task	UNCode
Task resolution	UNCode
Student grouping	UNCode, **UNColab**
Collaboration between students	**UNColab**
Collaboration assessment	**UNColab**, Google Forms

The functionality "Approaching the task" seeks that a teacher can propose a challenge or problem to solve, "Task resolution" considers a time where the student can solve the exercise individually, "Student grouping" establishes the use of a mechanism that allows assigning roles, "Collaboration between students" includes a moment in which a means of communication is provided to establish help and, finally, "Collaboration assessment" proposes an instrument that allows assessing the collaboration established in the exercise. As can be seen in Table 2, some functionalities required in the process are already provided by the UNCode platform; UNColab functions as a resource to support the collaboration and assessment processes. Consequently, the specifications for UNColab were defined based on the usage scenarios described below:

- Classify student: establish a division between students according to their success on the programming task.
- Assign group: assign a "Programmer" or "Novice" role according to the student's classification.
- Notify group: each student must know his/her role during the collaborative activity.
- Select partner: each student must be able to select a partner to establish a collaboration.
- Write comment: a communication mechanism should be provided to initiate a collaboration.
- Assess collaboration: each student should evaluate the collaboration.

2.2 UNColab development

Figure 1 shows the architecture of the proposed tool. As can be seen, two main components were established: a messaging component (Chat), which allows students to the message exchange, the role assignment, and the alert notification; and an API component, which facilitates communication between UNCode and the messaging component.

The "Chat" component works as a complement to the UNCode frontend, it has been developed using the React Js[2] interface library and the Firebase[3] platform for managing states, both messages and roles. On the other hand, the "API" component is developed in the Python language and is added as an extension of the INGinious[4] system, the base platform on which UNCode is developed.

[2] https://es.reactjs.org/

[3] https://firebase.google.com/

[4] https://inginious.org/

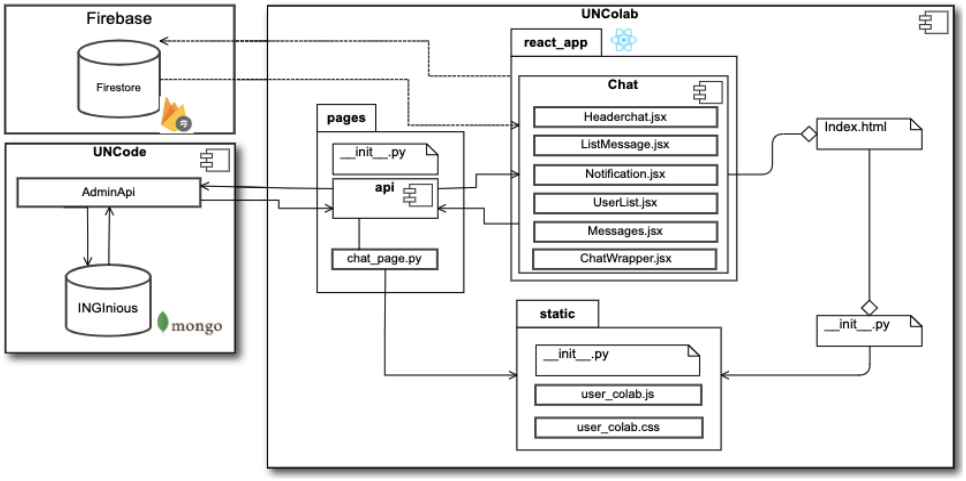

Figure 1. UNColab Architecture

Figure 2 shows the operation of the UNColab prototype jointly with the UNCode platform. As can be seen, the development is implemented according to the functionalities seen in Table 2, corresponding to the use scenarios proposed in the design. The grouping functionality is given by a pop-up window that notifies the status of the task and if the student has submitted at least two solution proposals in UNCode, according to the success rate of the task the system groups the student in "Programmer" or "Novice". The collaboration functionality allows the student, once grouped, to communicate with a collaborator from another group; in this case, through a chat mechanism. The evaluation functionality is given by a link that redirects to Google Forms looking for each student to evaluate the collaboration. The source code of UNColab is publicly available at: https://github.com/jhonmendex/un_colab

Figure 2. UNColab tool screenshots

3 Case study

3.1 Educational intervention

The collaborative learning activity was designed for students of the subject "Computer Programming" at the Universidad Nacional de Colombia. In the development of the topics of this subject, most of the proposed tasks are programming exercises, these seek that the student strengthens the concepts studied in the theory and can improve problem-solving skills through a programming language. The

UNCode platform supports the development, resolution and evaluation of these tasks, which is why it has been taken as a starting point in this study, since the automatic task evaluation functionality allows knowing the status of each of the submissions made by a student. Figure 3 presents the proposed flow of activities in the collaborative activity of this case study.

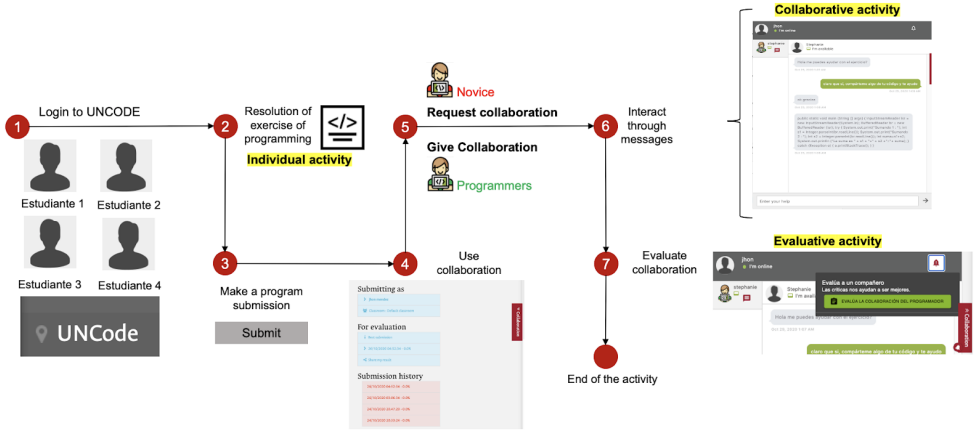

Figure 3. Activity flow in the collaborative activities of this case study

The flow starts with a student entering the UNCode platform (#1). Next, the individual activity (#2) is set up in which the student must solve a computer programming problem. The student is expected to submit a solution and have it evaluated by UNCode (#3). Once the submission activity is completed, the student can make use of the UNColab collaboration component (#4) and, depending on the outcome of the evaluated task, will have a role within the activity ("Programmer" or "Novice"); accordingly, help can be given or received (#5). From there, the Collaborative Activity (#6) begins, looking for students to exchange messages through UNColab. Finally, each student is expected to carry out the evaluation activity (#7) regarding the evaluation of the collaboration given or received.

This collaborative learning experience was conducted during two two-hour hands-on sessions in the computer programming class. The programming problems proposed for the collaborative activity addressed three basic programming topics: control structures, functions and arrays. Table 3 presents the distribution of activities in both interventions.

Table 3. Distribution of activities of the case study

	Intervention #1	**Intervention #2**
Topic	Control structures and functions	Arrays
Distribution of activities	• 20 minutes of individual activity • 25 minutes collaborative activity • 15 minutes of evaluative activity	• 20 minutes of individual activity • 25 minutes collaborative activity • 15 minutes of evaluative activity • 15 minutes to fill out a perception survey
Aspects of collaboration	• To request collaboration, a student must have at least two attempts to solve the assignment. • Once the student uses the collaboration option the system will notify if it is classified as "Programmer" or "Novice", according to his/her success rate in the exercise. • The system will notify a partner when the collaboration assessment should be done. • A "programmer" can help several "novices". • A "novice" can receive collaboration from various "Programmers". • The activity ends once the evaluative activity and the perception survey are completed.	

3.2 Participants y data collection

The study involved 27 first semester students of Computer Programming at the *Universidad Nacional de Colombia*. Both quantitative and qualitative data were collected in the two interventions. Regarding quantitative data, UNColab is able to collect data on:

- Number of “programmers”: number of students who solved the problem in the first two attempts.
- Number of “novices”: number of students who did not solve the problem in the first two attempts.
- Number of role changes: students who eventually converted from "novice" to "programmer".
- Number of collaborations: number of groups that were formed to collaborate.
- Number of messages: number of messages made per collaboration.

The qualitative data were collected at two different moments and referred to two aspects: the students' evaluation of the collaboration offered or received in solving programming problems and the general perceptions about UNColab. To know the evaluations about the collaboration offered and received among students, two surveys were created in Google Forms, corresponding to each of the roles that a student could have ("Programmer" or "Novice"). These surveys were attached to the tool as the final stage of the collaboration process. Table 4 presents the questions corresponding to the evaluation of the collaboration offered by the "Programmer" role, and Table 5 presents the assessment questions by the "Novice" role. In addition, perceptions about the UNColab tool were collected through a Google Forms survey, which was applied once the collaborative learning activity of the second intervention was completed. Table 6 shows the questions associated with the final perceptions survey.

Table 4. Collaboration assessment survey for "Programmers"

#	Question	Answer option
Q1	I consider that the collaboration given to my partner was useful to help him/her to improve his/her proposed solution to the programming problem.	*Likert scale* 1. Strongly disagree 2. Disagree 3. Somehow disagree 4. Somehow agree 5. Agree 6. Strongly agree
Q2	I consider that the collaboration given to my partner helped him/her to identify some errors in his/her proposed solution.	
Q3	I consider that the collaboration given to my partner helped him/her to correct some errors in his/her proposed solution.	
Q4	I believe that the collaboration provided allowed me to strengthen my knowledge about the topic of the problem.	
Q5	What opinion do you have about the collaboration offered to your partner?	Open-ended answer

Table 5. Collaboration assessment survey for "Novices"

#	Question	Answer option
Q1	I consider that the collaboration received allowed me to improve my proposed solution to the programming problem.	*Likert scale* 1. Strongly disagree 2. Disagree 3. Somehow disagree 4. Somehow agree 5. Agree 6. Strongly agree
Q2	I consider that the collaboration received helped me to identify some errors in my proposed solution.	
Q3	I consider that the collaboration received helped me to correct some errors in my proposed solution.	
Q4	I consider that the collaboration received gave me more confidence to try to solve the programming problem.	
Q5	What is your opinion about the collaboration received?	Open-ended answer

Table 6. UNColab students perception survey

#	Question	Answer option
Q1	I consider that the UNColab tool was useful for solving programming problems in a collaborative way.	*Likert scale* 1. Strongly disagree 2. Disagree 3. Somehow disagree 4. Somehow agree 5. Agree 6. Strongly agree
Q2	I consider that the UNColab tool facilitated the communication with my classmates to solve the programming problem in a collaborative way.	
Q3	I consider that the UNColab tool encouraged my participation in solving the programming problem in a collaborative way.	
Q4	In general, what do you think of the collaboration tool UNColab?	Open-ended answer

4 Results

According to the proposed activity flow, both interventions in the course establish an individual and a collaborative activity. The individual activity supported by UNCode made it possible to know which students managed to solve the exercises in the first two attempts and which did not. This information was later used by the UNColab component for the collaborative activity to perform the tasks of classifying students, forming groups and exchanging messages. Regarding this experience, it was identified in the first intervention that 13 students managed to solve the exercise in the first attempts without requesting collaboration; the tool classified them as "Programmers" and 14 students as "Novices". Once the collaborative activity started, 6 students managed to change their role from "Novice" to "Programmer". Similarly, in the second intervention, at the beginning of the activity, 12 "Programmers" and 15 "Novices" were identified; once the collaborative activity began, only 2 students were able to change their role from "Novice" to "Programmer".

During the collaborative activity, UNColab facilitated the formation of 12 collaborative groups or pairs for the first educational intervention and 39 for the second; it also allowed the exchange of messages

between these pairs during the time available for collaboration. Figure 4 shows the distribution of the number of messages made by each collaborative group in the two interventions carried out, as well as some atypical data related to the exchange of messages in intervention two during the collaboration time. A higher number of messages exchange is evidenced in the second intervention. The atypical data were due to groups or pairs of students who established constant communication throughout the collaborative activity, unlike most of the groups whose communication was sporadic. In this sense, in the messages of the first intervention, an average of 3.0 messages and a standard deviation of 2.9 were obtained. Regarding the second intervention, an average of 6.6 messages and a standard deviation of 11.0 were obtained.

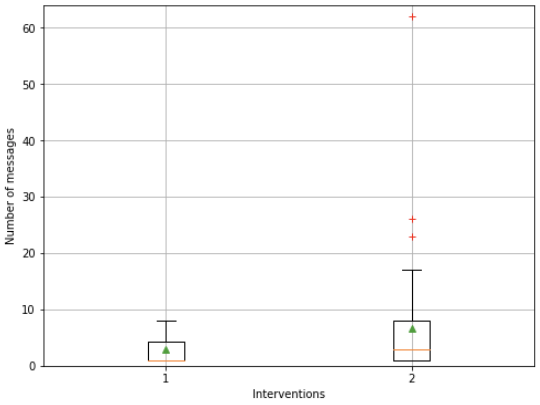

Figure 4. Box Plots of messages per collaboration and intervention

4.1 Collaboration assessment

The questions related to the instruments presented in Table 4 and Table 5 focused on the students' evaluations of collaboration when solving programming exercises from the point of view of the "Programmer" and the "Novice", respectively. Figure 5 shows the results obtained with respect to the first four questions of the evaluation instrument for "Programmers". Likewise, Figure 6 presents the results of the evaluation instrument for "Novices". The 100% of the students perceived with some level of agreement that the collaboration given or received had been useful to improve the solutions of the programming problems, to identify and correct errors in the solution, to consolidate the knowledge in the evaluated topic and to improve the confidence when solving the problem.

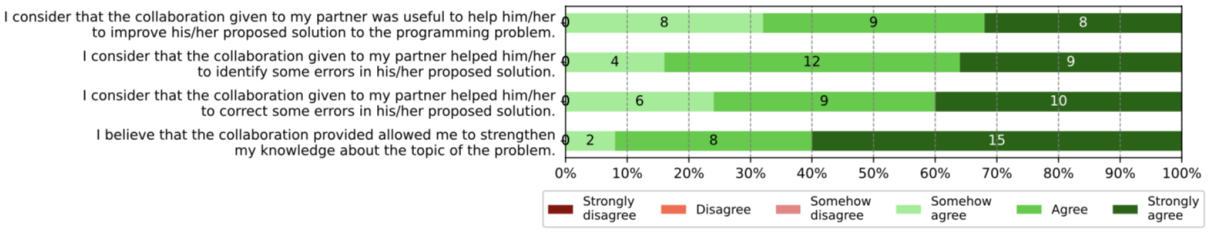

Figure 5. Collaboration evaluation results from "Programmers"

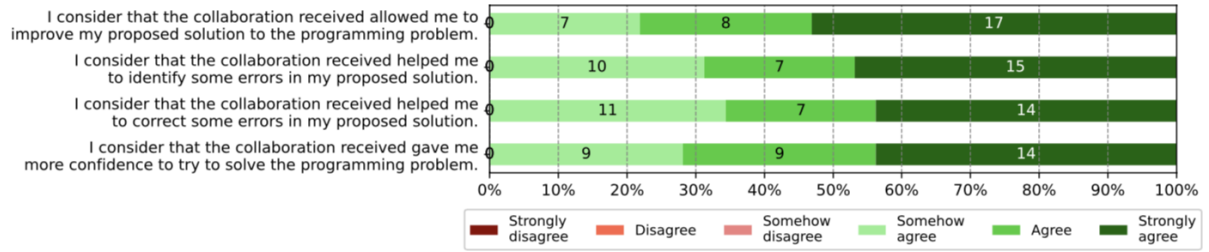

Figure 6. Collaboration evaluation results from "Novices"

The final question of each evaluation instrument sought to inquire more about the students' perceptions of collaboration, both in the role of "Programmer" and "Novice". Figure 7 presents the concepts related to collaboration in programming problem solving, where 5 main categories can be

observed. These categories were the result of several iterations of reviewing the students' responses, each response was classified and coded according to the interpretation and meaning of the message that the student expressed.

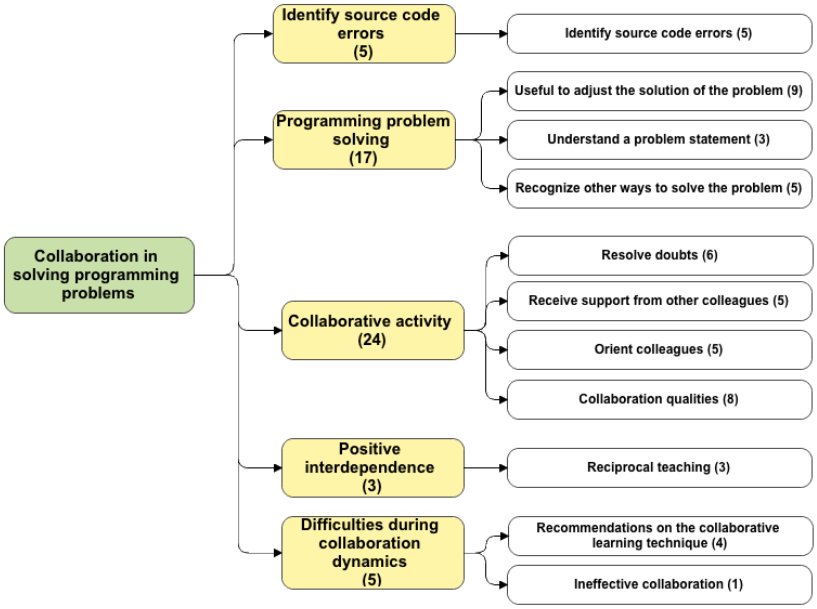

Figure 7. Perceptions from the students' assessment of collaboration

The category associated with error identification allowed grouping comments related to the benefit students see in receiving help from a peer to identify errors in the source code directly. "Programming problem solving" gathered comments associated with the usefulness of collaboration in understanding or finalizing an exercise. "Collaborative activity" focused on capturing perceptions about the main features and advantages of collaboration that students identified. "Positive interdependence" refers to perceptions involving reciprocal teaching; that is, students who stated that they learned something mutually through collaboration. Finally, "Difficulties during the collaborative dynamic" captured some comments related to inconveniences during the dynamic established for the study.

4.2 Students' perceptions when using UNColab

Figure 8 presents the results related to the first three questions of the perceptions instrument presented in Table 6. These questions were focused on knowing the evaluations related to the use of UNColab regarding its usefulness during the process of solving programming exercises through collaboration among students. More than the 92% of the students agreed with some level of agreement on the usefulness of the tool to solve problems collaboratively, the ease of communication between classmates for this purpose, and the utility of UNColab to encourage participation in collaborative problem solving.

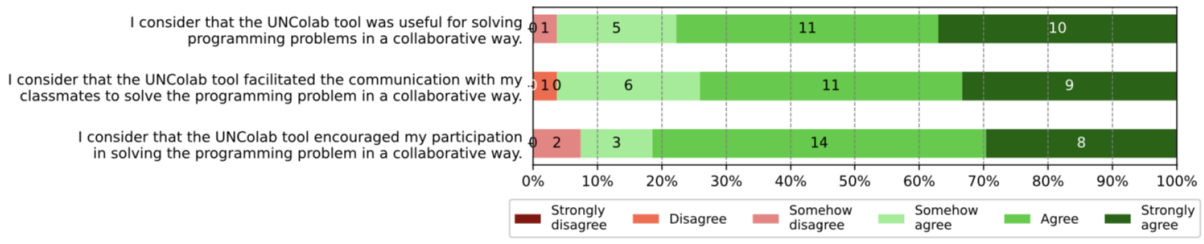

Figure 8. Students' perception with respect to the UNColab tool

The answers given by the students to the final question of the perceptions instrument (Table 6) resulted in 5 categories related to the use of UNColab (see Figure 9) as a collaborative learning tool for solving programming problems. For the definition of these categories, the same coding process mentioned in the previous section was carried out.

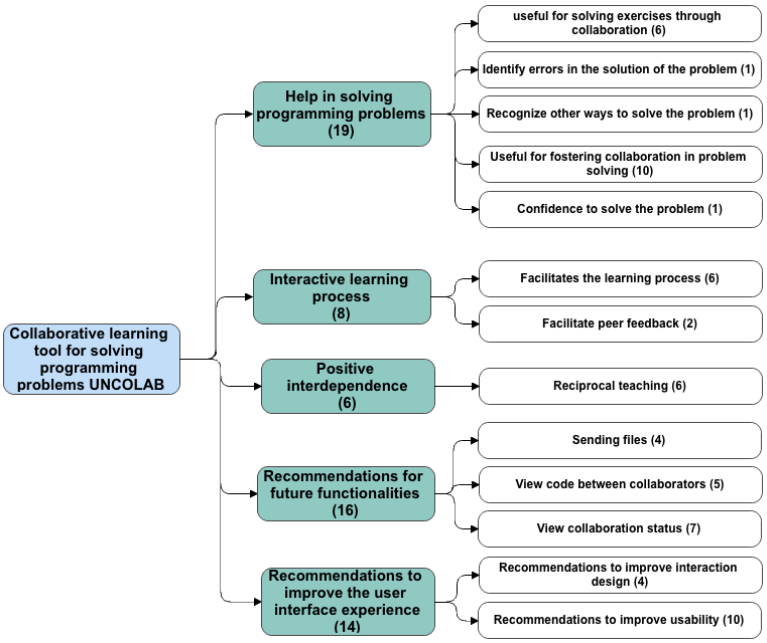

Figure 9. Student perceptions of UNColab

The category "Help in solving programming problems" relates the opinions about the usefulness of the tool as a facilitating element when requesting help. In "Interactive learning process", comments were collected regarding the recognition of the tool as an aid in the learning process. "Positive interdependence" gathers responses where a recognition of UNColab as a mediating element for reciprocal teaching is evident. "Recommendations for future functionalities" captures opinions about ideas for new functionalities that students expressed that could improve the tool during collaboration. Finally, the category "Recommendations to improve the user interface experience" relates aspects that can be improved to make the tool easier to use and have a better interaction.

5 Discussion

To answer the research question posed about the effects on programming problem solving when using UNColab, the results obtained at a quantitative and qualitative level were considered. The quantitative data collected on the use of UNColab indicate good results regarding the solution of the problems posed, especially for those students with the role of "Novice" who were able to solve each problem after the collaboration offered by a "Programmer". The data registered by the tool when used by the students show the importance of designing applications of this type using CSCL scripts that allow structuring the collaborative work. In this sense, it was possible to analyze the collaboration from the number of groups formed, the exchange of roles and the number of messages exchanged by the students, resulting in a complete participation of the course in the collaborative activity, achieving that at least each participant was able to establish communication with a peer and that in the second intervention some students made greater use of UNColab to establish constant communication with a collaborator at the time of solving the exercise.

We can highlight some benefits regarding the use of collaboration as a substantial element for solving computer programming problems. The qualitative data regarding the assessments of collaboration for both student roles ("Programmer" and "Novice") allowed us to identify and corroborate aspects related to the advantages of implementing collaborative activities within the classroom, such as those expressed in [3]. For example, the resolution of doubts, guidance and identification of errors are actions that in traditional contexts correspond properly to the teacher. The perceptions analyzed in this study show that these actions can also be reflected in students when collaborating.

As for the qualitative results, positive perceptions were revealed regarding the usefulness of this tool in the collaborative resolution of exercises, as well as a support for establishing communication among students and encouraging participation. In this sense, the results of the interventions showed that 44% of the participants believe that a collaborative activity can help in the resolution of doubts and concepts related to a collaborator's orientation. The 31% of the participants state that collaboration is beneficial

for the successful completion of a computer programming exercise. A 6% report opinions related to reciprocal teaching; which corroborates one of the advantages of the collaborative learning theory. Additionally, it is perceived that the tool developed makes the learning process more interactive and facilitates peer feedback. An important finding of these perceptions refers to future recommendations of new functionalities given by the students. Thanks to the suggestions received, it is possible to obtain the necessary requirements to build an application according to the needs of programming students, and to improve UNColab. This supports the idea presented in [17], which states that the use of prototypes is the best mechanism to identify software needs and thus to develop more robust tools.

In comparison with other studies, as for example [8], the collaboration between students was not done in a free way, instead the technique "Programmers and Novices" was used to form collaborative groups. This technique implemented in the tool played a differentiating role, since many of the students expressed the usefulness of knowing who can give or offer help at a given time.

6 Conclusions

This paper presented the computational tool UNColab, which supports the resolution of programming problems through collaboration among students. In addition, the experience of using UNColab in a computer programming course at the *Universidad Nacional de Colombia* was reported. The data provided by the tool, and the students' perceptions were analyzed. The main contributions of this study are the UNColab tool and the report of the experience obtained from its implementation in a computer programming course. The students' perceptions show that the use of this type of tools favors the teaching-learning of computer programming. UNColab focused mainly on the process of group formation and interaction through messages. However, this last aspect did not include an analysis of the quality of the interactions or their influence on the resolution of programming exercises; therefore, it is proposed as future work. In addition, some students suggested improvements and additional functionalities to UNColab, which will motivate the development of new prototypes.

REFERENCES

[1] K. M. Ala-Mutka, "A Survey of Automated Assessment Approaches for Programming Assignments," *Comput. Sci. Educ.*, vol. 15, no. 2, pp. 83–102, Jun. 2005.

[2] A. S. Marcolino and E. Barbosa, "A Survey on Problems related to the Teaching of Programming in Brazilian Educational Institutions," no. Icmc, 2017.

[3] Margarita Vinagre Laranjeira, "Teoría Y Práctica Del Aprendizaje Colaborativo Asistido Por Ordenador."

[4] L. M. Serrano-Cámara, et al., "MoCAS: A Mobile Collaborative Tool for Learning Scope of Identifiers in Programming Courses," *Int. J. Eng. Educ.*, vol. 32, no. 2, pp. 969–981, 2016.

[5] M.-J. Laakso, E. Kaila, and T. Rajala, "ViLLE – collaborative education tool: Designing and utilizing an exercise-based learning environment," *Educ. Inf. Technol.*, pp. 1–22, 2018.

[6] G. J. Hwang, Z. et al, "An Online Peer Assessment-Based Programming Approach to Improving Students' Programming Knowledge and Skills," *5th Int Conf Educ Innov. through Tech. EITT 2016*, pp. 81–85, 2017.

[7] E. S. J. De Faria, et al., "Forming groups for collaborative learning in introductory computer programming courses based on students' programming styles: An empirical study," in *Proc. Frontiers in Education Conf, FIE*, 2006, pp. 6–11.

[8] D. Tsompanoudi and M. Satratzemi, "A web-based authoring tool for scripting distributed pair programming," *Proc. - IEEE 14th Int. Conf. Adv. Learn. Technol. ICALT 2014*, Sect IV, pp. 259–263, 2014.

[9] O. Debdi, M. Paredes-Velasco, and J. Á. Velázquez-Iturbide, "GreedExCol, A CSCL tool for experimenting with greedy algorithms," *Comput. Appl. Eng. Educ.*, vol. 23, no. 5, pp. 790–804, Sep. 2015.

[10] L.-K. Soh, N. Khandaker, and H. Jiang, "I-MINDS: A multiagent system for intelligent computer- supported collaborative learning and classroom management.," *Int J Artif Intell Educ*, vol 18, no 2, pp 119-151, 2008.

[11] D. Hernandez-Leo, et al., "CSCL Scripting Patterns: Hierarchical Relationships and Applicability," in *Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06)*, 2006, pp. 388–392.

[12] F. Restrepo-Calle, JJ Ramírez-Echeverry, and F Gonzalez, "Uncode: Interactive System for Learning and Automatic Evaluation of Computer Programming Skills," *EDULEARN18*, vol 1, July, pp 6888–6898, 2018.

[13] E. D. Villasclaras-Fernández, et al., "Incorporating assessment in a pattern-based design process for CSCL scripts," *Comput. Human Behav.*, vol. 25, no. 5, pp. 1028–1039, 2009.

[14] E. F. Barkley, K. P. Cross, and C. H. Major, *Técnicas de aprendizaje colaborativo : manual para el profesorado universitario*. Ministerio de Educación y Ciencia, 2007.

[15] D. Hernández-leo and J. I. Asensio-pérez, "Generating cscl scripts," vol. 64, 2019.

[16] C. G. Hidalgo Suarez, V. A. Bucheli Guerrero, F. Restrepo Calle, and F. A. González Osorio, "Estrategia de enseñanza basada en la colaboración y la evaluación automática de código fuente en un curso de programación CS1," *Investig. e Innovación en Ing.*, vol. 9, no. 1, pp. 50–60, 2020.

[17] R. S. Pressman, Ingeniería del software: Un enfoque práctico. McGraw-Hill, 2010.