Case Study: A CSCL Approach for Object-Oriented Programming Learning

Leandro Flórez-Aristizábal[1](✉) and Fernando Moreira[2]

[1] Institución Universitaria Antonio José Camacho, Cali, Colombia
learistizabal@admon.uniajc.edu.co
[2] REMIT, IJP, Universidade Portucalense & IEETA, Universidade de Aveiro, Aveiro, Portugal
fmoreira@upt.pt

Abstract. A Computer-Supported Collaborative Learning (CSCL) approach was designed for the Systems Seminar course of the Systems Engineering program at University Antonio José Camacho (UNIAJC) in Cali (Colombia) [1]. The purpose was to make Object-Oriented Programming (OOP) learning not an individual process but a collective one where students play different roles to solve a software-based problem (from requirements specification to software development). Based on the results of the experts' review made in the previous study, a case study is now proposed to assess the feasibility of this approach. Eight students took part of this study following each of the stages proposed in the approach and the results show that even though this is still a good starting point, some changes need to be made to achieve better results.

Keywords: Computer-supported collaborative learning · Object-oriented programming

1 Introduction

Getting a group of people to work together, share understanding and achieve a common goal is a difficult task [2], especially for educators, since achieving true collaboration requires activities carefully designed for that purpose. Collaborative Learning (CL) is a strategy where a group of people learn together by interacting with each other and taking advantage of one another's skills and knowledge [3, 4]. Positive Interdependences are considered the 'heart' of collaboration [5] and they basically provide the elements to assure true collaboration among a group of learners. Some of these interdependences are: goals (group and individual), defining roles, sharing resources among team members, giving rewards for work done, creating identity for the team.

This paper presents a case study to validate a CSCL approach presented in [1] with eight students of the Systems Engineering program of the University Antonio José Camacho (UNIAJC) in a programming course, through interactive collaborative tools supported by computers. The paper is structured as follows: Sect. 2 shows related work on collaborative learning for programming courses. In Sect. 3, the CSCL proposal is presented. Section 4 presents the case study carried out, Sects. 5 and 6 show the results and discussion, finally Sect. 6 concludes the study.

T. T. Primo et al. (Eds.): WAVE 2021, CCIS 1425, pp. 55–64, 2022.
https://doi.org/10.1007/978-3-031-07018-1_5

2 Background and Related Work

Several studies discuss about how students in programming courses not always achieve expected results [6–8] and this may be the consequence of multiple reasons such as lack of motivation or not appropriate teaching/learning methods [9]. Collaborative Learning could fill this motivational gap by allowing students to learn from each other as a group. For this research, the following studies were taken as a starting point for the design and implementation of an approach presented in [1].

An Object-Oriented Programming (OOP) Course was redesigned in [10] were knowledge mediation was carried out through a system called ViLLE, a collaborative education tool that allow students to work on different type of programming problems. While students work, the systems automatically assess their work and give proper feedback when it is submitted.

The course was redesigned to promote collaboration, active learning and to facilitate communication between students and their teacher. This system also allowed integrating surveys in a different way that allowed learners to identify visual, auditory, and communicative variables related to the process of OOP learning. The software also provided tutorials as didactic material and the student has the opportunity to express concerns about the progress with the system.

The results obtained through this system show that the implementation of the evaluations through learning environments favors communication. Researchers were also able to integrate writing skills to be able to compile the code and test it to identify mistakes. The main goal was to promote collaboration among students by using learning environments that allow cognitive, communicative, and technological development in the learning process related to OOP. The main component of the study is that it supports the development of collaborative learning as a process of educational innovation. The number of students that passed the course increased by more than 20% in both instances of the redesigned course according to the authors of the study. This study shows one way to redesign a typical programming course, making good use of available technology to promote collaboration.

Beck and Chizhik [11], review the principles of cooperative learning, and describes how these principles were incorporated into a comprehensive set of cooperative learning activities for a CS1 course. In the activities carried out, roles were assigned to the members of the group for individual accountability.

The group processing is followed by a whole-class debriefing led by the instructor, which works in tandem with the group activity to help students improve their understanding of the material. The effectiveness of these cooperative learning activities was assessed in a series of educational research studies which spanned three academic years and included two different instructors. The results of these study showed statistically significant benefits from the cooperative learning approach, both overall and for a broad range of subgroups of students.

This study gives outstanding examples of how students can collaboratively help others understand the process of software development and this was considered for our proposal.

3 Computer-Supported Collaborative Learning Proposal for OOP Teaching

The CSCL proposal of this study is aimed at students from the Systems Engineering program at university Antonio José Camacho in Cali (Colombia). The course is Systems Seminar where students learn the basics of software engineering (algorithms, system requirements, class diagrams and object-oriented programming). In this course, the students must follow a series of steps to give possible solutions to common problems.

To make this work collaborative, Positive Interdependencies (PI) were considered to assure collaboration among students. The PIs that were chosen for this approach are: Roles, Resources, Identity, Goal, Tasks, and Reward (This is the grade for the activity). Some strategies proposed by Kaila et al. (2016) were also considered in the design of this approach.

3.1 Defining Roles

First, students are split into groups of 4 people and all of them must choose a name that identifies them as a team. Each team member is given a role and their correspondent responsibilities/resources. The roles are explained in detail in the following section.

Role 1: Requirements Specifier (RS). This student is in charge of defining the requirements of the system. The client (teacher) gives the information needed for this work. The student is allowed to ask as many questions as necessary to gather and specify the requirements of the system.

Role 2: Class Diagrammer (CD). This student receives the requirements and analyzes them with the rest of the group to make the necessary changes (add/remove/modify the requirements proposed by the RS). Then the CD designs the class diagram of the solution.

Role 3: Algorithm Designer (AD). This student analyzes the class diagram along with the rest of the group to make the necessary changes to it (add/remove/modify the classes proposed by the CD). Then, the AD designs the algorithms of the methods defined in the class diagram.

Role 4: Solution Developer (SD). This student analyzes the algorithms along with the rest of the group to make the necessary changes to the algorithms. Then, SD develops the solution in Java programming language.

The stages to be followed are shown in Fig. 1.

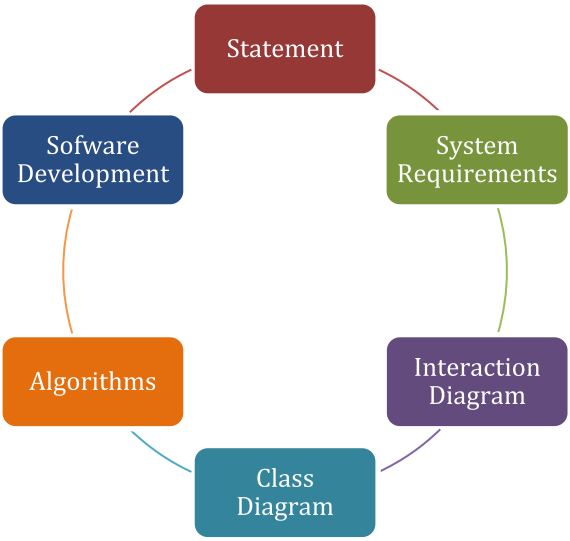

Fig. 1. Stages of the proposal

Each role has its purpose in each stage. Sometimes students will work on their own playing their roles in the corresponding stage with the opportunity to discuss the work done with the rest of the team. The Interaction Diagram stage is the only one with no specific role for it, instead, all team members will work together to define the interaction between objects. The idea behind this proposal is to let students build knowledge in the whole process of developing software, from requirements to the final product. It is important to highlight that changes proposed by the experts' in the review done in [1] will be considered for the case study.

4 Case Study

4.1 Methods

Participants. Eight students at University Antonio José Camacho (Cali, Colombia) were invited to be part of this case study. According to the proposal previously mentioned, each team should have 4 students, each one taking one of the roles proposed for the activity. One teacher was in charge of moderating the activity and his role was to explain what the students had to do, how much time was going to be invested and the tasks assigned to each role.

Students' Profile.

- Age: All students are between 20 and 23 years old.

- Gender: 3 Girls and 5 boys
- All students are in 3rd semester of the systems engineering program.

Environment. The activity was carried out in a virtual environment through the ZOOM platform due to the available features such as breakout rooms to separate students in isolated virtual spaces and thus ensure that they do their work individually. It also offers text, video, and voice interaction.

Programming Language. The software must be developed in Java language since this is the language they are learning in the current semester. The user interface (UI) is managed using the JOptionPane package through pop-up dialogs given that they have not yet learned how to create their own UIs.

Problem Statement. A small store is having problems with their inventory because all the information about products is being kept in a notebook (prices and available units), and this information must be updated every time a unit is sold or if the price changes. Sometimes, due to the amount of people buying in the store, the person in charge does not have enough time to write down the information about all the sells made, so the notebook remains out of date. A simple system is required so that information about prices, available units of each product and sells can be easily updated or viewed through a computer.

Setup and Execution. This process guarantees individual accountability and support from peers in every stage of the process. This process should be repeated interchanging roles to ensure that all students develop or strengthen different skills, unfortunately, it was not possible for the 8 students to be in this activity for 4 consecutive weeks to play all the roles due to their different commitments with the regular semester and their jobs.

This proposal is based on a 3-h class, and two teams of 4 students were created, each student with a role. Then, the requirement specifiers (1 per team) were isolated in a virtual classroom and only the teacher was allowed to talk to them and answer the questions they ask. This first stage (system requirements) lasts 12 min (Fig. 2).

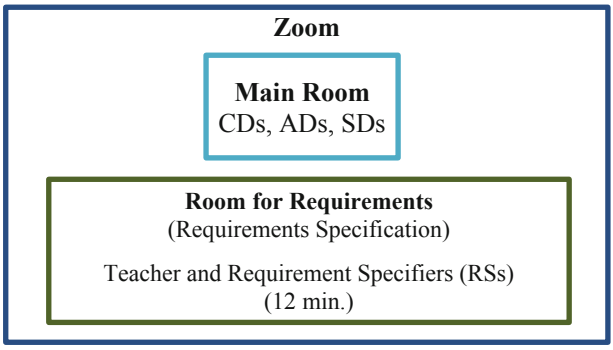

Fig. 2. Requirements specification stage

Once each RS defined the requirements for his/her team, they were called to the main virtual classroom. Virtual spaces were created (one for each team). The four members of each team were invited to join their virtual space to discuss the requirements for 8 min (Fig. 3).

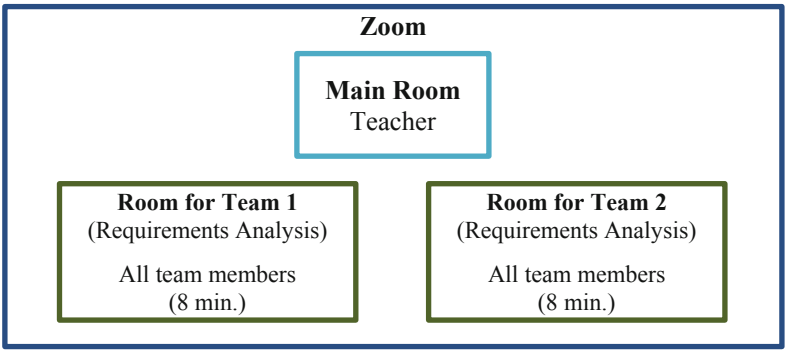

Fig. 3. Requirements analysis

With the requirements specified, all team members remained in their virtual room and defined the Interaction Diagram of the solution. 15 min were given for this stage.

After that, the RS, AD (algorithm designer) and SD (solution developer) left the room and only the CD had the task to design the class diagram of the solution (17 min). If the CD needed help (with conceptual elements of the diagram), only the teacher could provide it (Fig. 4).

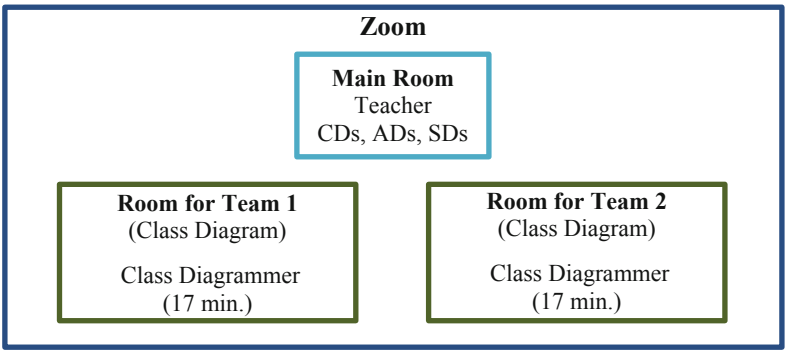

Fig. 4. Class diagram stage

When the CD's of each team finished their diagrams, the rest of the team members joined the separated virtual space to discuss the diagram for a period of 8 min. After that time, only the AD stayed in the room to design the algorithms (35 min) while the rest of the team members returned to the main classroom. If the AD needed help (with conceptual elements of the algorithm), only the teacher could provide it (Fig. 5).

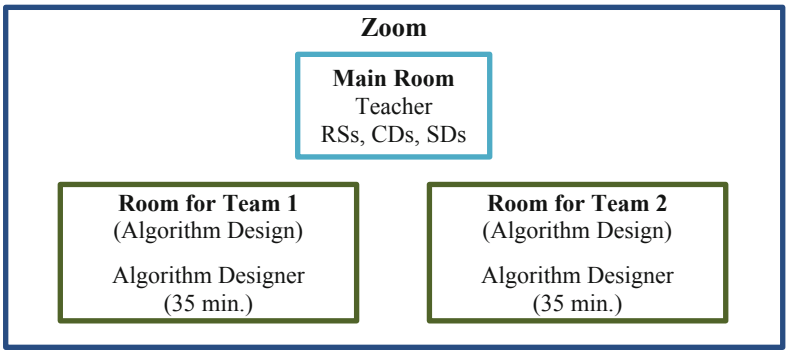

Fig. 5. Algorithm design stage

Finally, all team members joined the AD in the team's virtual space to discuss the algorithms for a period of 15 min. When the discussion ended, only the SD stayed in the team's virtual room to develop the software solution according to the class diagram and the algorithms designed in previous stages (Fig. 6).

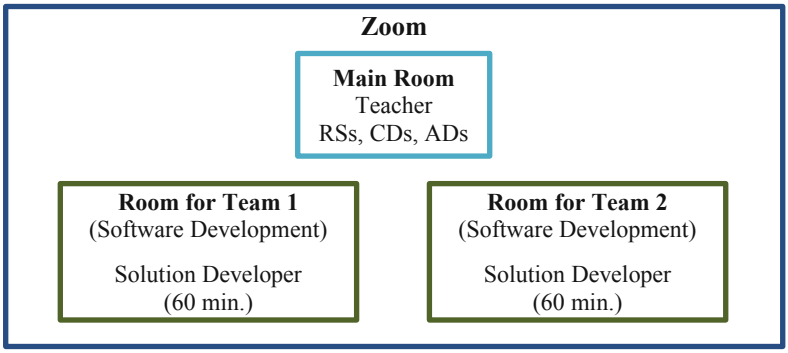

Fig. 6. Software development stage

In this stage, the SD had the possibility to ask for help. All members of the team could join the SD for a maximum of 3 min to solve issues with the software development, then, the RS, CD and AD had to leave the room. If the problem was not solved, the SD could ask for help from the teacher. Help from team members could be requested 3 times maximum (3 min per request). The development of the software could not last more than 1 h.

The remaining 25 min of the class were invested in analyzing the solutions of each team and discussing the problems faced in the design process.

5 Results

During the activity, the role of the teacher as moderator was very important to control the time during the activities and provide help when needed. Students from Team 1 were very agile during the activity, they even finished some stages before the estimated time, while students from Team 2 had some troubles not only at an academic level but also at a technological level because 2 of the team members were having serious problems with their internet connection, so they could not perform their role as expected, this led to change some of the rules for this team and provide more time to finish some stages.

At the end of the activity, during the analysis of both solutions, the students of Team 1 presented a well-designed solution with minor flaws. The solution presented by students from Team 2 had some major issues due to not all students were at the same academic level and some of them needed more time with their team mates to clear their doubts, but for the purpose of the activity, and with so limited time for it, it was not possible to give them more time.

Both teams presented:

- Enough and well written requirements
- Interaction diagram (Team 2 had minor problems with it)
- Class diagram (Team 2 had some conceptual doubts about it that were resolved during the analysis phase with all team members)
- Algorithms for some methods
- A software based on pop-up dialogs (Team 2 had an issue that could not be resolved within the given time and sometimes the application was unexpectedly closed).

Before ending the activity, students were asked the following open questions:

- What do you think about breaking down the activities, so each student is in charge of a particular task?
  - o All 8 students agreed that having different stages and roles for it strengthen their skills on a particular stage of the process, so they all think that this is something valuable for the learning process.
- What would you change about the overall activity?
  - o 4 out 8 students think that the time for algorithm design and software development is too long for those who are not in charge of these activities, they felt like they wanted to do more (requirement specifiers and class diagrammers) but all they could do was wait for the solution developer to request for help.

- Do you think this method could improve your understanding of the whole process of software development? Why?

  - All 8 students agreed this could improve their learning methods since they count on their team mates to support the work they are doing and not rely only on their knowledge but everyone's in the team. 5 students expressed their willing to be part of the whole process (4 weeks) in the future.

6 Discussion

Having the teacher moderate the activity is very important to control time, clear doubts and make students feel there is someone backing up the process.

While this proposal is promising to take advantage of virtual education, these activities mediated by technology may be interrupted by technical problems like those experienced by Team 2.

The results showed that dividing the work by stages and roles, help students master skills and share knowledge with their peers. Although, the time spent by those not working on algorithms or programming, must be rethought to take advantage of it. Before carrying out a new case study, it would be important to improve this approach with these results and opinions given by students.

7 Conclusions and Future Work

This case study was proposed to continue the approach presented in a previous study. Taking advantage of the current situation where some classes are still being held in virtual environments, a group of students were invited to take part of this study to validate a CSCL approach for OOP learning.

First, all recommendations given by experts in the previous study were taken and applied for this case study, such as decrease time for some stages and adding the Interaction Diagram stage.

The case study showed that working in the development of a simple software-based system as a team can be beneficial for learning OOP and strengthen individual and collective skills during the process, understanding that each team member has individual accountability, and the efforts of all members are necessary to achieve a common goal. The implementation of positive interdependences made it easier to guarantee true collaboration.

While the results of the case study are positive, there are still improvements to be made and more case studies must be carried out to evaluate the process of the learners performing different roles each time. Case studies with in-person classes are also necessary to guarantee that it can be beneficial also without virtual environments.

References

1. Flórez-Aristizábal, L., Burbano, C.L., Moreira, F.: Towards a computer-supported collaborative learning approach for an object-oriented programming course. In: Rocha, Á., Adeli, H., Dzemyda, G., Moreira, F., Ramalho Correia, A.M. (eds.) WorldCIST 2021. AISC, vol. 1367, pp. 163–172. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-72660-7_16
2. Häkkinen, P., Järvelä, S.: Sharing and constructing perspectives in web-based conferencing. Comput. Educ. **47**(4), 433–447 (2006). https://doi.org/10.1016/j.compedu.2004.10.015
3. Guerrero, L.A., Mejías, B., Collazos, C.A., Pino, J.A., Ochoa, S.F.: Collaborative learning and creative writing. In: Proceedings of First Latin American Web Congress (LA-WEB 2003), p. 7 (2003)
4. Bagheri, S., Rostami, N.P., Pour Kivy, S., Lahiji, E.R.: Collaborative learning, collaborative teaching & autonomy : a survey study on English as a second/foreign language. Mod. J. Lang. Teach. Methods **5**(3), 348–356 (2015)
5. Laal, M.: Positive Interdependence in collaborative learning. Procedia - Soc. Behav. Sci. **93**, 1433–1437 (2013). https://doi.org/10.1016/j.sbspro.2013.10.058
6. Hanks, B., McDowell, C., Draper, D., Krnjajic, M.: Program quality with pair programming in CS1. ACM SIGCSE Bull. **36**(3), 176–180 (2004). https://doi.org/10.1145/1026487.1008043
7. Jenkins, T.: On the difficulty of learning to program. In: Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences, pp. 53–58 (2002)
8. Wu, P.: Practice and experience in the application of problem-based learning in computer programming course. In: Proceedings of ICEIT 2010 - 2010 International Conference on Technology for Education, vol. 1, no. Iceit, pp. 170–172 (2010). https://doi.org/10.1109/ICEIT.2010.5607778
9. Nikula, U., Gotel, O., Kasurinen, J.: A motivation guided holistic rehabilitation of the first programming course. ACM Trans. Comput. Educ. **11**(4), 1–38 (2011). https://doi.org/10.1145/2048931.2048935
10. Kaila, E., Kurvinen, E., Lokkila, E., Laakso, M.-J.: Redesigning an object-oriented programming course. ACM Trans. Comput. Educ. **16**(4), 1–21 (2016). https://doi.org/10.1145/2906362
11. Beck, L., Chizhik, A.: Cooperative learning instructional methods for CS1: design, implementation, and evaluation. ACM Trans. Comput. Educ. **13**(3), 1–21 (2013). https://doi.org/10.1145/2492686