F. I. Anfurrutia, A. Álvarez, M. Larrañaga and J. -M. López-Gil, "*Visual Programming Environments for Object-Oriented Programming: Acceptance and Effects on Student Motivation*," in **IEEE Revista Iberoamericana de Tecnologias del Aprendizaje**, vol. 12, no. 3, pp. 124-131, Aug. 2017, doi: 10.1109/RITA.2017.2735478

Visual Programming Environments for Object Oriented Programming: Acceptance and Effects on Student Motivation

Felipe I. Anfurrutia, *Member*, *IEEE,* Ainhoa Álvarez, Mikel Larrañaga, *Member*, *IEEE,* Juan-Miguel López-Gil

***Abstract*—This paper presents an experiment in which visual programming environments have been used in programming courses aiming at helping students to acquire the competencies of a course on Object Oriented Programming. The study presented is centred on the analysis of the acceptance by the students of this type of environment, as well as its effect in the motivation of the students. The results obtained show differences in their answers according to the two possible characteristics of the students analysed: their gender and the fact of being a newcomer or retaking the subject.**

***Index Terms*—Computer science education, Object Oriented Programming**

I. Introduction

In Computer Science degrees, the technical skills associated with programming are covered by several modules. In the Bachelor Degree In Computer Engineering in Management and Information Systems at the University of the Basque Country (UPV/EHU), two modules –*Introduction to Programming* and *Modular and Object Oriented Programming*– address the main specific competencies of programming in the first course. Acquiring these competencies is essential to obtain good results in many of the subjects of this degree.

Although students are aware of their relevance in the degree, these modules have high dropout and failure rates, thus posing a challenge for both teachers and students.

One of the main issues of these modules is an increase in students of the programming modules who present a high degree of heterogeneity regarding previous knowledge of the competencies taught in these modules [1], [2]. This issue makes it difficult for teachers to design appropriate learning methods for all students [5].

Programming modules are usually taught using general purpose programming languages that may be very complex for students [1], [3]. Some programming languages require students to learn many concepts before they can begin any programming task, while others require typing large amounts of code, which are difficult to understand by novice students. This means that students have to deal with both the construction of the algorithms and the syntactic rules of the programming languages used.

Some authors propose the use of visual programming environments as they reduce the cognitive overload implied in the initial programming tasks of any programming module. Although they do not solve syntax related problems, they allow the postponement of the syntactical peculiarities and also to focus the students initially on concept understanding or designing tasks without worrying about the syntax [2], [4]. Once they have understood the basics, they can move on to a non-visual environment and address the problem of code syntax.

Our hypothesis is that the use of visual programming environments in programming subjects can improve the learning experience of students. However, the mere fact of including new support tools does not solve the problems the majority of students face. In order to provide adequate pedagogical support, the use of support tools must be meticulously designed.

This article first presents a proposal for teaching improvement by incorporating visual programming environments. Subsequently, the implementation of the proposal in an object-oriented programming subject is described. Next, the study carried out is presented, followed by the analysis and discussion of the results obtained. Finally, some conclusions and future lines are mentioned.

II. Proposal for Teaching Improvement

The quality of learning and teaching programming improves by using constructivist approaches, in which students construct knowledge actively instead of being mere passive recipients of such knowledge [5]-[9]. Therefore, discovery and experience can promote learning in these subjects. That is why, so far, the teachers of the programming modules have tried to redesign their lessons in order to make them as practical as possible. To this end, an attempt has been made to expand the number of examples and practical exercises. However, the efforts made have not been successful enough to improve the academic results. One of the reasons is that the practical exercises have been done using a professional programming environment, Eclipse, which forces students to simultaneously deal with the complexity of abstract concepts, the syntax of the programming language and the complexity of the environment itself.

F. I. Anfurrutia, University of the Basque Country UPV/EHU, felipe.anfurrutia@ehu.eus
A. Álvarez, University of the Basque Country UPV/EHU, ainhoa.alvarez@ehu.eus
M. Larrañaga, University of the Basque Country UPV/EHU, mikel.larranaga@ehu.eus
J-M. López-Gil, University of the Basque Country UPV/EHU, juanmiguel.lopez@ehu.eus
DOI (Digital Object Identifier) Pendiente

Any educational innovation should be based on learning theories. In the case of programming, practical situations are the most appropriate [8]. Among the different theories proposed for experimental learning, Kolb's learning cycle stands out [10]. This learning cycle has often been used in programming learning processes [11], [12]. It entails four stages in which students must get involved in order to acquire knowledge (see Fig. 1). First, they must carry out a specific activity. Afterwards, they should reflect on the experience, to later conceptualize the theoretical aspects that allow the explanation of the observed behaviour. Finally, they must apply the theory in new situations.

The appropriate application of this learning cycle implies combining it with the use of tools that promote or facilitate identified cognitive processes (observation/reflection, conceptualization, application and experimentation). These tools include, for example, visual programming environments or educational robots. As the use of educational robots presents the problems inherent to the use of physical devices, the authors decided to use the visual programming environments in this case. The use of these environments will allow the "Concrete experience" to be graphically addressed in the computer laboratories and, then, the abstract conceptualization phase to be performed more appropriately under the guidance of the teaching staff.

The next section depicts how this learning cycle has been applied in the module in charge of introducing the Object Oriented Programming (OOP) paradigm.

III. Implementation of the Proposal

A. Subject Contents

Modular and Object Oriented Programming (MOOP) is a compulsory module of the first year of the degree in Computer Engineering in Management and Information Systems at the UPV/EHU. This module is the second one related to the subject of programming among the students' study, since the *Introduction to Programming* module has been taken by all students in the first term. Therefore, the students who take this module already have knowledge about the code syntax and basic concepts of programming in Java, but they lack OOP knowledge.

In the MOOP module, four main topics are addressed (see Fig. 2): Fundamental OOP concepts (classes, objects and methods), interactions (i.e., message passing between objects), inheritance and unit testing. In addition, students should also be able to understand the UML class diagrams at the end of this course.

The educational innovation carried out entails the application of Kolb's learning cycle to each topic of the module. The next subsection describes the tools chosen to support this methodology.

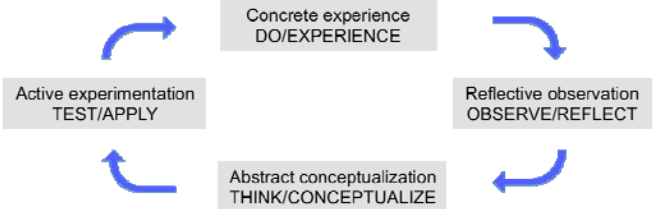

Fig. 1. Kolb's learning cycle

B. Tool Selection

When considering the introduction of new programming environments, it must be taken into account that: 1) there is no programming environment that is suitable for all situations [3], and 2) the activity performed by using a tool must be attractive and relevant for students [11].

There are several tools that support OOP learning. Among them, BlueJ and Greenfoot are two of the most complete and advantageous ones [13], and are better suited to Kolb's learning cycle [11]. Both tools have different levels of complexity and support different parts of the module. Therefore, the authors decided to use both, as shown in Figure 2, in this module. BlueJ was mainly used to introduce the fundamental concepts during the first 7 weeks, whereas Greenfoot was used to work on interaction and inheritance aspects in the remaining 8 weeks. Both tools were used two hours per week in computer laboratories.

The main features of each tool are described next.

1) BlueJ

BlueJ is an integrated development environment (IDE) for learning OOP with Java language designed for educational purposes [14]-[16]. The distinguishing feature of BlueJ is its graphical user interface (see Fig. 3), which in the upper part shows the class diagram in UML-like format, and at the bottom the object bank containing the instances created. Through this visualization, students can inspect both classes and objects. Moreover, they can interactively create objects and invoke methods without writing any code or having deep knowledge of Java. This way, students can experiment and reflect with the concepts of class and object before reaching the state of conceptualization.

In addition, it also allows students to introduce some necessary syntax elements, since students can view and edit the code for a specific class when they double-click on it.

2) Greenfoot

Greenfoot is another integrated development environment (IDE) created to help students learning OOP related aspects [11], [17]. This development environment allows interactive applications in two-dimensional worlds to be created. Its rich visual scenarios allow learning and teaching based on simulations or games [12].

In relation to MOOP themes, it helps to introduce the concept of inheritance interactively in two ways: by defining subclasses and inspecting both inherited attributes and methods (see Fig. 4). As Greenfoot is based on BlueJ, inspections are carried out in a familiar way for students, because they have previously used BlueJ.

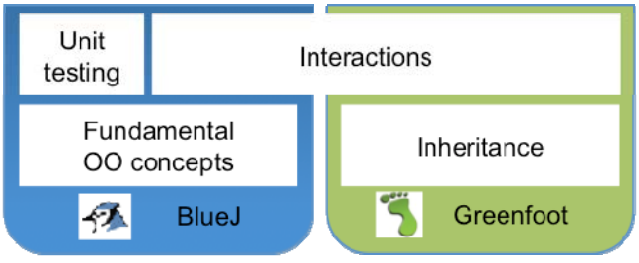

Fig. 2. Subject Contents

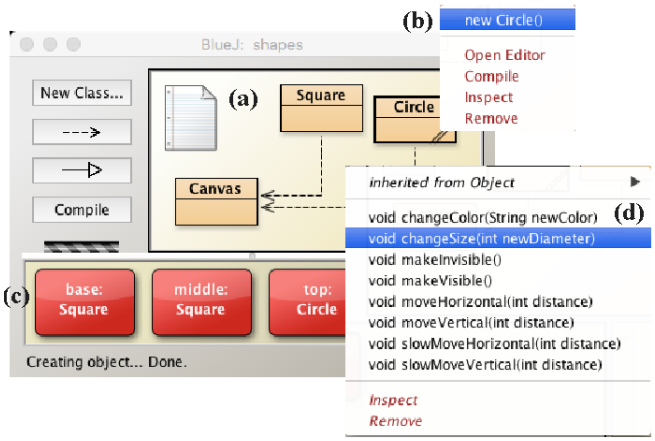

Fig. 3. BlueJ integrated development environment: a) class diagram, b) instantiation process, c) instantiated objects and d) method call

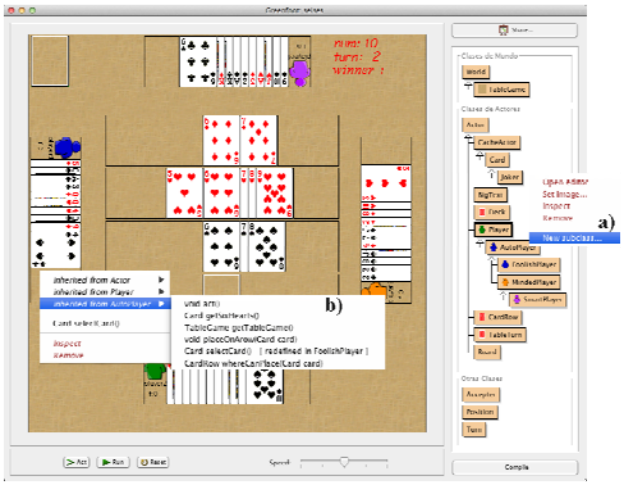

Fig. 4. Greenfoot integrated development environment: a) definition of a new subclass and, b) inspection of inherited methods

C. *Kolb's Learning Cycle Application Example*

All the topics of the module were tackled by means of the Kolb's learning cycle with the aim of promoting active learning and using the visual tool selected for each topic.

Figure 5 shows how Kolb's learning cycle was applied using BlueJ to address the "Fundamental Concepts of OOP" topic:

1. **Concrete experience**: Students were provided with a project that included a set of classes representing geometric shapes (e.g., Circle, Square) which could be drawn on a canvas. The specific tasks students had to perform included creating instances of those classes, drawing them on the canvas, moving them around, and changing their size or color by message passing. To this end, they interacted with classes and objects using their methods (see Fig. 3b and d).
2. **Reflective Observation**: Students had to observe the effect of their actions by inspecting the state of the objects both before and after the message passing to an object (method execution). For example, one of the tasks involved drawing two squares and a circle of different sizes (50, 40 and 30 respectively), one on top of the other. In each of the tasks, students should reflect on a table each of the performed operations. Tasks also included questions to guide students through the reflective process, helping them to understand the effect of the performed operations.
3. **Abstract Conceptualization**: Teachers guided this phase during the lectures. They explained the notions of class, object and method conceptually. The explanations were based on the projects used in the computer laboratories.
4. **Active Experimentation**: Students were presented with a new scenario in which they had to apply the learned concepts. In particular, students had to define new classes and objects, and interact with them to draw what was requested by means of geometric figures (see Fig. 5).

Once these phases were completed, the students worked on the syntax aspect. First, they inspected the code of a specific class and then they implemented a program using the code presented in the table of the reflective observation phase (see Fig. 5-2).

Greenfoot has been used in a similar way using various scenarios proposed in [18], such as *Leaves and Wombats*, *Little Crab*, and others specifically created by the teachers.

IV. Study

A. *Objetives*

The general objective of the proposal described in section III is to improve the performance of students in programming modules by incorporating visual learning environments. In order to improve this, they need to perceive the pedagogical benefits of the proposal and improve their initial motivation. Based on the results obtained in previous studies in the *Introduction to Programming* module, the following specific objectives are proposed for this study:

- Objective 1: Study of how the use of visual programming environments affects the initial motivation and expectations of students.
- Objective 2: Analysis of the help provided by visual environments in the acquisition of the concepts addressed in a subject in which OOP concepts (in this case MOOP) are taught.
- Objective 3: Study whether the programming environments used are suited to the contents of the module.
- Objective 4: Study of how the use of visual environments within the framework of Kolb's learning cycle affects the performance of students.

Taking the results of previous studies into account, for each objective of this work, the differences between the students will be analyzed based on: (a) being retakers –those who are repeating the module– and non-retakers and (b) gender.

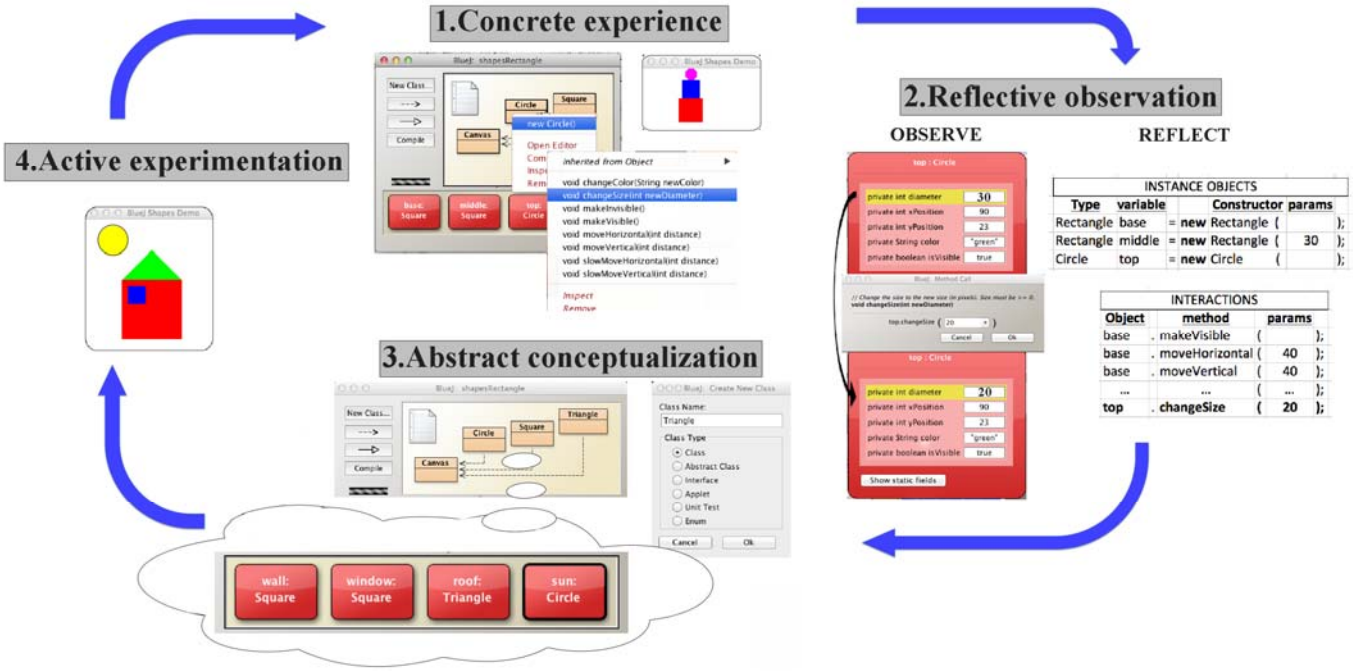

Fig. 5 Example of Kolb's learning cycle application using BlueJ for the "Fundamental concepts of OO" topic

B. *Data Collection*

Different sources of information were used in this study. For the first three objectives, data was collected through a survey that was conducted before the final test and the publication of the final marks of the course. This data was collected during the three academic courses after the implementation of the methodology. For the fourth objective, the final marks of the students in the module were considered. In the latter case, the teachers of the module considered that using two different methodologies with two groups of students when one produces better results than the other was not appropriate [15]. Therefore, a between-subject analysis was conducted, in which the students are not divided into two different groups, but rather their results in different courses are compared. For this aim, the results of the three academic courses prior to the implementation of the methodology and the three subsequent ones, in which it was already implemented, were used.

The survey, in addition to questions for contextualization data, contained a set of five-level Likert-type questions (see Table I): strongly disagree, disagree, indifferent, agree, and strongly agree. The questions were grouped by each objective. Moreover, for each block, an open-ended question was included to collect other comments that the students wanted to convey about the experience. Table I shows an excerpt from the survey completed by the students of the module.

A total of 56 students who completed the course did this survey. Its results are detailed below.

V. Results and Discussion

The results obtained for each of the objectives analyzed in this study are described below. Regarding the contextualization of the results, of the total of 56 students, 21% were female and 79% were male. On the other hand, 70% of students were non-repeaters.

TABLE I: EXCERPT FROM THE SURVEY

Objective	Code	Question
Objective1	BlueJ.Motivation	My motivation in the subject has increased with the use of BlueJ
	Greenfoot.Motivation	My motivation in the subject has increased with the use of Greenfoot
Objective2	BlueJ.Helpful	Using BlueJ has helped me with the module
	Greenfoot.Helpful	Using Greenfoot has helped me with the module
Objective3	BlueJ.No	I would have preferred not to use BlueJ
	Greenfoot.No	I would have preferred not to use Greenfoot
	Eclipse	I would have preferred to use Eclipse

A. *Objective 1: Motivation*

Concerning student motivation in using these environments, only 23% of all students think that using BlueJ did not increase their motivation; In the case of Greenfoot, this value is reduced to 16% (see Fig. 6). There was a large undecided percentage when answering this question, which in the case of BlueJ reached 52%. This high level of indecision fell to 32% with Greenfoot, implying that 52% of students are very clear about the increase in their motivation using this tool.

Comparing the results between retakers and non-retakers, differences are observed in the case of Greenfoot. While 23% of non-retakers indicate that using Greenfoot did not increase their motivation, none of the retakers responded in this sense. The percentage of students who acknowledge that the use of Greenfoot has positively influenced their motivation is similar in both groups (53% in the case of retakers versus 51% in non-retakers) (see Fig. 7).

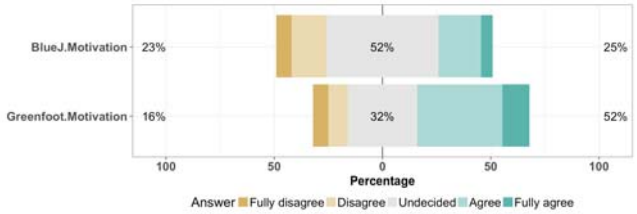

Fig. 6. Motivation increase using BlueJ or Greenfoot

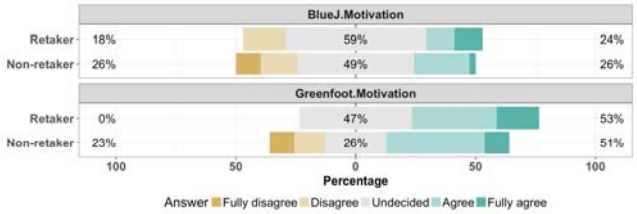

Fig. 7. Motivation increase using BlueJ or Greenfoot, grouped by retakers and non-retakers

Analyzing the data collected for Greenfoot, which more clearly affects the motivation of students in a positive way, the gender difference is remarkable (see Fig. 8).

57% of non-retaking female students think that the tools used did not increase their motivation in the subject, while in the case of male retakers, this percentage is reduced to 16%. The 80% of indecision among female retakers is also remarkable, which falls to 33% in the case of male students.

As expected, Greenfoot game-based scenarios generally improve the initial motivation of students. However, this effect does not appear to be the same between male and female students. This circumstance, in agreement with those of [19], [20], seems to indicate that the scenarios are not equally attractive to females or males and it is an aspect that should be analyzed in depth. The positive results of the experiences presented in [21] suggest as a possible way to follow the design of diverse visually attractive scenarios with a specific set of problems that better suit the tastes of the students.

B. *Objective 2: Help to Understand*

Students were asked, for each of the used tools, whether they had helped them to assimilate the concepts of the module or not, and how much it had helped them in each of the topics covered.

In this respect, only 14% of the students consider that the use of these environments has not helped them (see Fig. 9).

It is remarkable that the percentage of students who think that the visual tools have helped them is higher among the retakers (65%), as opposed to 46% of the non-retakers, as can be observed in Fig. 10.

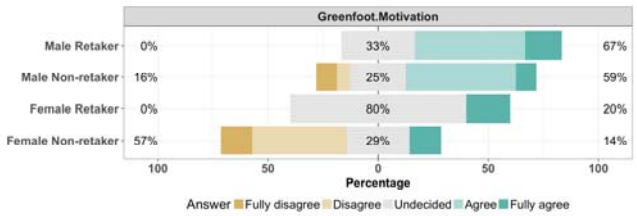

Fig. 8. Motivation increase using Greenfoot, grouped by gender and retakers/non-retakers

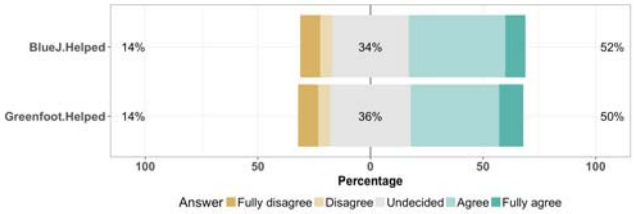

Fig. 9. Those who think that using BlueJ or Greenfoot helps to understand

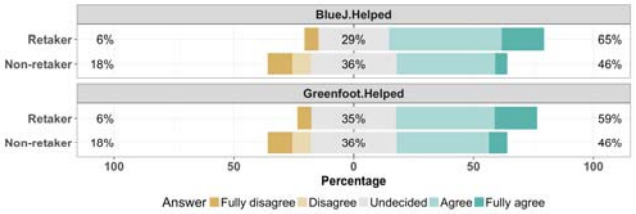

Fig. 10. Those who think that using BlueJ or Greenfoot helps to understand, grouped by retakers/non-retakers

Analyzing the differences by gender (see Fig. 11), the results are worse among female students. While 20% of the female retakers have responded negatively to this question, not one of the male retakers has issued negative responses to it. In addition, indecision is also greater among females than among males. The correlation observed between the results associated with this objective and objective 1 may suggest that, in the case of females, these negative results are related to the type of scenarios that had been used.

C. *Objective 3: Tool Suitability*

Considering the tool selection, students were first asked if they would have preferred not to use BlueJ or Greenfoot (see Fig. 12). 16% of students would have preferred not to use BlueJ and 11% not to use Greenfoot. However, the percentage of those who would like to keep the tools is clearly higher than those who do not (54% Greenfoot and 39% BlueJ).

Comparing the results obtained and separating them into retakers and non-retakers, among the retakers, the results are better with Greenfoot. Only 12% of them would have preferred not to use it, while 65% clearly indicated its convenience (see Fig. 13).

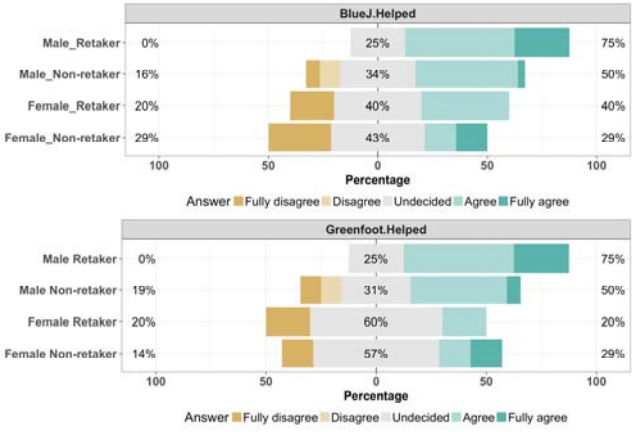

Fig. 11. Those who think that using BlueJ or Greenfoot helps to understand, grouped by gender and retakers/non-retakers

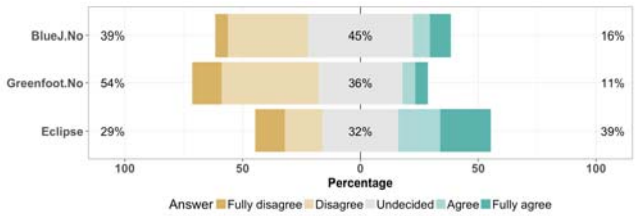

Fig. 12. Those who would have preferred not use BlueJ or Greenfoot, or just using Eclipse

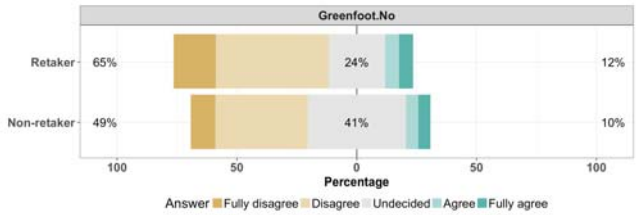

Fig. 13. Those who would have preferred not to use Greenfoot grouped by retakers/non-retakers

In the survey, students were also asked if they would have liked to work with Eclipse and not use visual learning environments. 53% of the retakers indicated that they would have liked to use Eclipse exclusively, despite the utility they see in used visual environments (see Fig. 14). However, among non-retakers, the percentage is lower (33%) and the level of indecision is higher among non-retakers (36% vs. 24%).

Analyzing the open question related to this objective, it can be detected that interest in Eclipse is because it offers more functionality and is perceived to be more useful for the professional future of the students. However, the fact that retakers are less reluctant to use educational tools suggests that, despite the limitations in this aspect presented by the tools, they do appreciate that they contribute significantly to learning.

When analyzing the results from a gender perspective, on the one hand, the considerable indecision of females in the case of BlueJ is noteworthy. This is particularly noticeable in the case of retakers, where the rate of indecisiveness rises to 80%, as opposed to 17% among male repeaters (see Fig. 15, upper part). In the case of Greenfoot, the value rises to 71% in the case of female non-retakers compared to 34% in the case of male non-retakers (see Fig. 15, lower part). On the other hand, it is remarkable that the majority of students in favor of maintaining visual environments are male: 67% of retakers in the case of BlueJ and 41% of non-retakers; in the case of Greenfoot, the value increases to 75% of retakers and 56% of non-retakers (see Fig. 15, lower part).

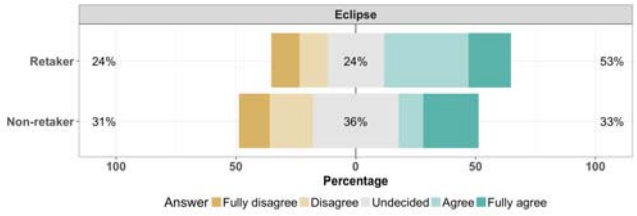

Fig. 14. Those who would have preferred to use Eclipse, grouped by retakers/non-retakers

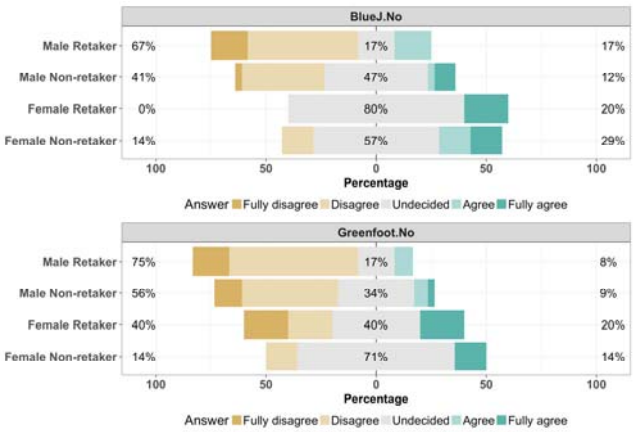

Fig. 15. Those who have preferred not to use BlueJ or Greenfoot, grouped by gender and retakers/non-retakers

Similarly, most of the female students (both retakers and newcomers) indicated clearly that they would have preferred to start working exclusively with Eclipse (see Fig. 16).

D. Objetivo 4: Improvements in the Grades

In order to determine whether the educational innovation carried out has helped to improve the students' grades, the academic results of the last three courses before implementing the new methodology (C1, C2, C3) and the three subsequent courses (C4, C5, C6) have been analyzed. Due to their special characteristics, the third and fourth years (C3 and C4 respectively) are considered as transition years. The third one because it correspondes to a change of curriculum and, therefore, students were more motivated to pass and not be forced to change the academic plan, while the fourth was the first year in which the new methodology was implemented.

Concerning the percentage of students presented, the rate went from less than 40% in the courses prior to the change of methodology to an average of 55% of those presented. This data by itself is very interesting, although the improvement of the grades of the students who were submitted to the final exam is also noteworthy.

As can be seen in Fig. 17, the percentage of students who passed the exam increased from around 45% to around 70%.

Students who take the exams are usually those who are confident that they will pass them. The increase in this number suggests that the motivation and confidence of the students has improved with the new methodology (Objective 1). In addition, improved results support the idea that the use of new tools within Kolb's framework actually helps students in their learning process (Objective 2).

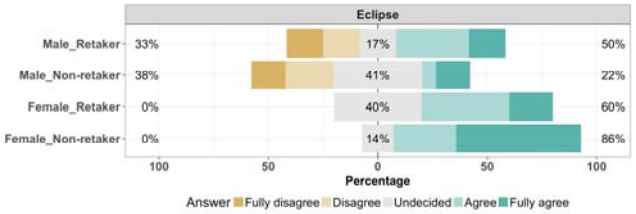

Fig. 16. Those who would have preferred to use Eclipse instead of the other environments, grouped by gender and retakers/non-retakers

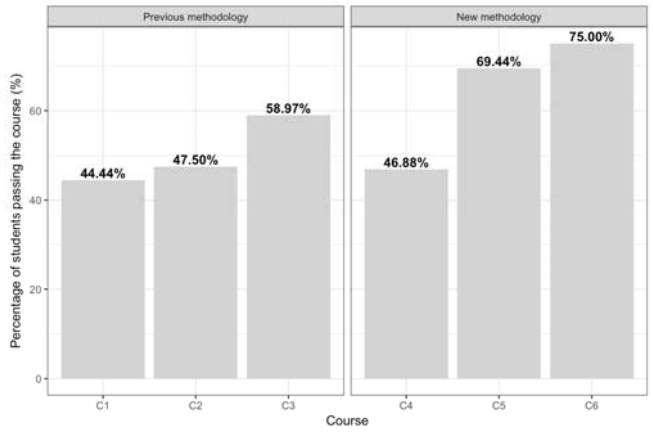

Fig. 17. Percentage of students who passed the exam

VI. Conclusions and Future lines

This paper has presented an implementation of Kolb's learning cycle through the use of visual learning environments in the setting of the teaching of the OOP paradigm. This learning cycle has been implemented for each of the topics during three academic courses. Its implementation has been supported by two different visual environments: BlueJ and Greenfoot.

The study of the results has been performed analyzing, from the perspective of the students, how the new proposal affected their motivation, whether it helped them in their learning or not and whether the tools fit the syllabus of the module or not.

As for motivation, overall, the results are not as good as had been expected; in fact they are worse than those previously obtained with physical robots in the *Introduction to Programming* module [3], [22], [23].

In relation to the help they provide, the results have generally been positive. In addition, students have indicated in the surveys that they would not stop using the tools chosen. However, it is worth mentioning the great influence of the gender of the students in the result of the surveys. Female's answers are noticeably more negative than male's in this regard.

On the other hand, a large percentage of students suggest that they would like to work directly with Eclipse. One of the main reasons for this is the functionality provided (e.g., code templates, refactoring, etc.). Visual educational environments include less functionality and are not as efficient for the development of more complex applications. To continue using these kinds of tools in a more adequate way, it seems appropriate to remind students that the aim of the subject is to learn concepts, rather than a specific programming language or IDE.

Based on these results, a set of future lines are opened in order to continue with the implementation of improvements in the programming modules.

On the one hand, considering that previous experiences with robots in other modules reflect a higher degree of motivation and that grades improve more with visual environments, adequately combining both tools could be an interesting way to improve motivation without affecting the performance.

On the other hand, it would also be appropriate to better integrate the visual programming environments in the evaluation process of the module or to relate them to other modules in order to provide the students with a global perspective. Hence, it is also suggested that it would be interesting to first apply the learning cycle in *Introduction to Programming* and later in *Modular and Object Oriented Programming*. This could make the students see them integrated in a better way.

Using the theory of self-determination [24] as a framework to measure student motivation could also be considered.

Finally, the great differences in the results according to gender raise an urgent need to expand the study before continuing with new implementations. For example, whether the problem lies in the selected tools or the subject matters of the performed exercises, particularly selected scenarios should be analysed. According to [19], [20], the perception of computing, and games in particular, greatly varies according to gender. Expanding this part of the study would allow us to adapt the improvements that have been proposed in this work.

Acknowledgements

This work is supported by the Basque Government (IT980-16), the University of the Basque Country UPV/EHU (EHUA16/22, GIU16/15, and PIE 6819).

References

[1] A. Gomes and A. J. Mendes, "Learning to program-difficulties and solutions," in *International Conference on Engineering Education–ICEE*, Coimbra, Portugal, 2007, vol. 2007.

[2] D. J. Malan and H. H. Leitner, "Scratch for budding computer scientists," *ACM SIGCSE Bulletin*, vol. 39, no. 1, pp. 223–227, 2007.

[3] A. J. Hirst, J. Johnson, M. Petre, B. A. Price, and M. Richards, "What is the best programming environment/language for teaching robotics using Lego Mindstorms?," *Artif Life Robotics*, vol. 7, no. 3, pp. 124–131, Sep. 2003.

[4] A. Wilson and D. C. Moffat, "Evaluating Scratch to introduce younger schoolchildren to programming," *Proceedings of the 22nd Annual Psychology of Programming Interest Group (Universidad Carlos III de Madrid, Leganés, Spain*, 2010.

[5] D. C. Leonard, *Learning theories, A to Z*. Westport, Conn.: Oryx Press, 2002.

[6] S. Papert, *The children's machine: rethinking school in the age of the computer*. New York: BasicBooks, 1993.

[7] C. Wang, L. Dong, C. Li, W. Zhang, and J. He, "The Reform of Programming Teaching Based on Constructivism," in *Advances in Electric and Electronics*, W. Hu, Ed. Springer Berlin Heidelberg, 2012, pp. 425–431.

[8] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen, "A study of the difficulties of novice programmers," *ACM SIGCSE Bulletin*, vol. 37, no. 3, pp. 14–18, 2005.

[9] F. Jurado, A. I. Molina, M. A. Redondo, and M. Ortega, "Cole-Programming: Shaping Collaborative Learning Support in Eclipse," *Tecnologias del Aprendizaje, IEEE Revista Iberoamericana de*, vol. 8, no. 4, pp. 153–162, Nov. 2013.

[10] D. A. Kolb, *Experiential learning: experience as the source of learning and development*. Prentice-Hall, 1984.

[11] P. Henriksen and M. Kölling, "Greenfoot: combining object visualisation with interaction," in *Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, 2004, pp. 73–82.

[12] L. Yan, "Teaching Object-Oriented Programming with Games," in *Sixth International Conference on Information Technology: New Generations, 2009. ITNG '09*, 2009, pp. 969–974.

[13] S. Georgantaki and S. Retalis, "Using educational tools for teaching object oriented design and programming," *Journal of Information Technology Impact*, vol. 7, no. 2, pp. 111–130, 2007.

[14] D. J. Barnes and M. Kölling, *Objects first with Java: a practical introduction using BlueJ*. Boston: Pearson, 2012.

[15] M. Kölling, "Using BlueJ to introduce programming," in *Reflections on the Teaching of Programming*, Springer, 2008, pp. 98–115.
[16] M. Kölling, B. Quig, A. Patterson, and J. Rosenberg, "The BlueJ System and its Pedagogy," *Computer Science Education*, vol. 13, no. 4, pp. 249–268, 2003.
[17] M. Kölling, "The Greenfoot Programming Environment," *ACM Transactions on Computing Education*, vol. 10, no. 4, pp. 1–21, Nov. 2010.
[18] M. Kölling, *Introduction to Programming with Greenfoot: Object-Oriented Programming in Java with Games and Simulations*, 2nd ed. Pearson, 2016.
[19] J. Robertson, "The influence of a game-making project on male and female learners' attitudes to computing," *Computer Science Education*, vol. 23, no. 1, pp. 58–83, Mar. 2013.
[20] M. . Phan, J. R. Jardina, and W. S. Hoyle, "Video Games: Males Prefer Violence while Females Prefer Social," 2012. [Online]. Available: http://usabilitynews.org/video-games-males-prefer-violence-while-females-prefer-social/. [Acceded: 19-Jun-2016].
[21] R. B. Hijon-Neira, Á. Velázquez-Iturbide, C. Pizarro-Romero, and L. Carriço, "Game programming for improving learning experience," in *Conference on Innovation & technology in computer science education ITiCSE '14*, 2014, pp. 225–230.
[22] C.-C. Wu, I.-C. Tseng, and S.-L. Huang, "Visualization of Program Behaviors: Physical Robots Versus Robot Simulators," in *Informatics Education - Supporting Computational Thinking*, R. T. Mittermeir and M. M. Sysło, Eds. Springer Berlin Heidelberg, 2008, pp. 53–62.
[23] A. Álvarez and M. Larrañaga, "Experiences Incorporating Lego Mindstorms Robots in the Basic Programming Syllabus: Lessons Learned," *Journal of Intelligent & Robotic Systems*, vol. 81, no. 1, pp. 117–129, Jan. 2016.
[24] R. M. Ryan and E. L. Deci, "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being," *The American Psychologist*, vol. 55, no. 1, pp. 68–7. 2000.

Dr. Felipe I. Anfurrutia is a lecturer at the UPV/EHU and a member of the research group ONEKIN Web Engineering (http://www.onekin.org). His research lines are related to the development of Web applications through Domain Specific Languages, Software Product Lines, and different programming paradigms (OOP, Features, Aspects and XML). He has also participated in educational projects to improve teaching in programming.

Dr. Ainhoa Álvarez is a lecturer at the UPV/EHU in the Department of Computer Languages and Systems. She works in the field of educational computing in the GaLan research group (http://galan.ehu.eus). Her main lines of research focus on the analysis of learning and teaching assisted by technology in engineering.

Dr. Mikel Larrañaga is a lecturer at the UPV/EHU in the Department of Languages and Computer Systems. He works in the field of educational computing in the GaLan research group. His interests include knowledge acquisition, knowledge maps, intelligent tutors, learning analysis and technology-assisted teaching in engineering.

Dr. Juan Miguel López Gil is a lecturer at the UPV/EHU and member of the GaLan research group. He has worked on usability and accessibility of user interfaces, adaptive user interfaces and emotional computing.