Incorporating educational robots and visual programming environments in introductory programming courses

Felipe I. Anfurrutia, Ainhoa Álvarez, Mikel Larrañaga and Juan-Miguel López-Gil
Department of Languages and Computer Systems, University of the Basque Country, UPV/EHU,
Nieves Cano 12, 01006, Vitoria-Gasteiz, Spain.
{felipe.anfurrutia, ainhoa.alvarez, mikel.larranaga, juanmiguel.lopez}@ehu.eus

***Abstract*— Introductory programming courses are very challenging both for students and teachers. Several authors propose incorporating either educational robots or visual programming environments as a means to ease learning of programming. This paper presents three experiences that use these kinds of tools with the aim of applying Kolb's experiential learning cycle. The experiences have been carried out in the first year courses of programming in the *Bachelor in Computer Management and Information Systems Engineering* at the Faculty of Engineering of Vitoria-Gasteiz in the UPV/EHU.**

Keywords—Educational Software, experiences of use, programming

I. INTRODUCTION

One of the main problems in introductory programming courses is the great heterogeneity regarding the students' previous mastery level of the competences tackled in those courses [1]. This fact hampers the design of adequate learning methods for the whole group of students [2]. Moreover, programming courses are usually taught using general-purpose programming languages, which sometimes become very complex to novice students without previous knowledge [1], [3]. Some programming languages have a very steep learning curve whilst others require writing a high amount of code, which newcomers can hardly understand and tackle, even for the simplest programs. In general, students have to deal at the same time with the design of the algorithms and the syntactic rules of the programming languages being used.

Two main options are proposed in the literature to lighten the problems novice students face in programming courses [4]. On the one hand, visual programming environments have been used. These environments allow students isolating from the complexity of general-purpose programming languages, allowing them to focus on the comprehension of the fundamental concepts before beginning to program [5]. On the other hand, the use of physical devices has also been explored. This approach allows students to program those devices and interact with them in real world scenarios [6].

The authors of this paper have explored both approaches in the last few years. However, the mere fact of introducing those elements in a course does not imply any educational improvement *per se* [7]. Therefore, their introduction in the course has been done in different degree of combination with Kolb's learning experiential cycle [8]. This study has been carried out during the last 5 academic years in the two courses related to programming in the first year of the *Bachelor in Computer Management and Information Systems Engineering* at the Faculty of Engineering of Vitoria-Gasteiz: Introductory Programming (IP) in the first term and Modular and Object-Oriented Programming (MOOP) in the second term.

The paper is organised as follows. First, the general design of the experiences is described, followed by the description of the specific experiences carried out. Next the results obtained from the analysis of the experiences are presented. Afterwards, some aspects derived from the experiences that should be considered when using any of the proposed options are detailed. Finally, the conclusions of the work are presented.

II. GENERAL DESIGN OF THE EXPERIENCES

The authors of this paper have carried out several experiences in the last few years with both educational robots and visual programming environments. For both options, a study of the available environments and devices has been performed in order to select the more adequate one attending to the specific characteristics of the courses and the proposed pedagogical improvement [3], [9], [10].

Only including the new tools does not imply educational improvement; any implementation must be carefully designed [7]. In the experiences presented in this paper, Kolb's experiential learning cycle has been applied in different degrees [8]. This methodology has four stages (see Figure 1) in which students must actively participate in order to acquire the knowledge. First, they must develop a specific task. Next, they must reflect on the experience in order to later on be able to conceptualise the theory that allows explaining the observations. Finally, they must apply the theoretical foundations in new situations.

This work has been supported by the Vicerrectorado de Estudios de Grado e Innovación of the University of the Basque Country (UPV/EHU) who through the SAE/HELAZ has supported the PIE 6819 project in the biennium 2014-2016.

978-1-5090-4596-9/16/$31.00 ©2016 IEEE

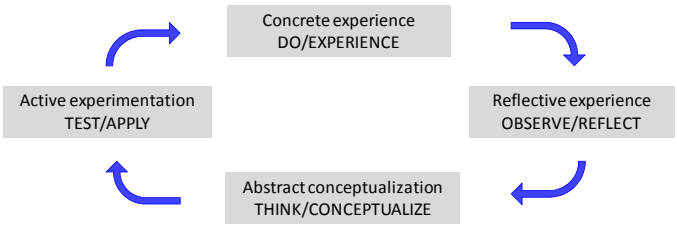

Fig. 1. Kolb's experiential learning cycle

In order to evaluate the adequacy of the experienced innovations, the perception of the actors involved and the evolution of the learning marks have been analysed. To this end, both qualitative and quantitative studies have been carried out. The information has been collected by means of surveys that students had to fill in order to assess the tools and their use in the learning of the course concepts.

III. EXPERIENCES CARRIED OUT

The three experiences are described next.

A. *First experience*

The objective of this experience was to help first year students dealing with one of their main problems: Algorithm design. In particular, the main goals of this experience were to improve the students' problem solving abilities and to introduce the design related concepts. Those concepts are treated during the first lessons of the course.

For this experience, educational robots where selected, more specifically, Lego MINDSTORMS® educational robots. These robots are provided with a visual programming environment based on blocks. This environment was used expecting that the students would explore and experiment with the different types of blocks and understand their semantics. Furthermore, the students were expected to conceptualize the semantics and, then, relate them with the algorithmic notation and elementary programming concepts. Those robots where used in the IP course during three academic years with 100 students [10], [11].

B. *Second experience*

After three years using educational robots in IP, the teachers of the subject decided to extend the application of Kolb's experiential learning cycle to the whole course. However, some authors have evidenced negative experiences when the robots are used throughout the whole semester. In general, students have limited access to the robots out-of-school and this limitation interrupts their positive impact [12].

Therefore, robots where rejected and the Scratch visual programming environment was selected (*https://scratch.mit.edu/*) for this second experience. Although Scratch is oriented to a younger audience, it has been also used for the teaching of programming in several educational centres and even in universities [13]. Scratch has been used in the IP course during the last two academic years, because it allows students to deal with the main programming concepts (sequence of sentences, conditional sentences and so on), isolating students from the syntactical aspects of general-purpose programming languages. Around 110 students took part in this experience. The lessons in which Scratch was used where organised as follows:

1. Scratch was used to introduce a concept (e.g., conditional statements) through a solved example
2. Students solved in Scratch an exercise, similar to the previously presented one
3. Students represented the program developed in the exercise of the previous step using algorithmic notation
4. Students implemented the exercise using the Java language and the Eclipse programming environment

C. *Third experience*

The third experience was implemented in the Modular and Object-Oriented Programming (MOOP) course. Students arrive to this course in the second semester after having taken IP, so they already have the basic notions of algorithms. The main objective of this experience was to help students understanding concepts related to Object Oriented Programming (OOP). Again, visual programming environments were chosen. These environments allow students to address and graphically observe each concept in the 'concrete experience' and 'reflective experience' stages, of Kolb's experiential learning cycle respectively. Then, teachers can guide students in the 'abstract conceptualization' stage. Finally, students can also visually test/apply each concept in new cases in the last stage, i.e. 'active experimentation'.

BlueJ (*http://bluej.org*) and Greenfoot (*http://greenfoot.org*) were chosen for this course; both visual development environments have been designed with pedagogical purposes in the learning of OOP [14], [15]. These environments provide students with visual tools, such as the UML class diagrams for the object representation or viewers to analyze the state and behaviour of objects. They also provide students with shortcuts (without having to write any line of code) to either create objects or interact with these objects (message passing). During the execution of these tasks, these environments guide students with implementation issues (introducing the code syntax and leaving blanks for students to fill in).

Both visual programming environments have been used in the MOOP course during five academic years with 340 students. Due to their characteristics, from the 15 weeks the course consist of, *BlueJ* was used during the first half of the course to introduce the concepts of class and object, whereas in the second half *Greenfoot* was used to work with the concepts of inheritance and polymorphism. The last tool was also used out-of-school time to develop a final course Project.

Moreover, at the end of each laboratory session, students had to answer a test related to the laboratory statement and the concepts treated in the laboratory session, which better allowed integrating the used tools and the learning method in the evaluation of the course.

IV. RESULTS OBTAINED IN THE STUDY OF THE EXPERIENCES

The main results of each of the presented experiences are described next.

A. First experience

From the analysis of the answers to the survey, it can be derived that both the interest and the motivation of students increased due to the use of robots. Regarding the learning awareness of students, which is essential in any learning process, students perceived that robots helped them to better understand the concepts tackled in the course. The course teachers, who had also detected an increment of students' motivation and an improvement in the class atmosphere, corroborated these observations. It was very stimulating for teachers seeing how some students recorded and distributed videos in which robots were executing some tasks. Despite these positive effects, there was no statistically significant improvement on the marks of the students.

There were also some negative aspects. On the one hand, despite being conscious of the positive effects of robots, some students asked to directly use a general-purpose environment such as Eclipse. This can be due to the fact that the robots were only used in the first lectures and their use was not evaluated later. These results are in line with those of [16], which describe that students prefer to eliminate those activities that have not direct impact in the marks. On the other hand, some difficulties inherent to the use of physical devices, such as aspects related to the movement of robots or to the light conditions of the laboratories were also observed. The different friction of surfaces or the charge of the batteries affected the movement of the robots, whereas the different light conditions, even inside the same room, affected the recognition of the colours, essential for the development of certain tasks.

B. Second experience

As the semester progressed, it was observed that students explored and experimented with Scratch. They tested alternatives to the proposed exercises or even tried to develop additional activities. Regarding the evaluation, some questions about Scratch (comprehension or blocks or recognising errors or adequate solutions) were also included, providing better integration with the course syllabus and the students being less reluctant to use it.

However, as Scratch is designed for a younger audience, some students found it too childish.

C. Third experience

According to the studies developed, students consider that the tools employed helped them in the learning process. Students have also indicated that they would like to continue using them. However, opinions differ according to the gender of the students. The results of the survey showed that women have notably worse opinion than men.

The academic results of the students have also been analysed. With the inclusion of these tools to support Kolb's experiential learning cycle, the percentage of sitting students which was around 40%, has increased to a rate over a 60% Moreover, the percentage of sitting students that pass the course has increased from 45% to rates close to 70%.

However, the results regarding the motivation results do not fulfil the expectations of the involved teachers. In fact, the results are worse than those previously obtained with educational robots in the IP course [4], [10]. A high indecision was observed, especially in the case of BlueJ, which reached 52% of the students.

V. Discussion

Based on the results of the experiences, a set of considerations to be taken into account before implementing any similar experience in programming courses is outlined.

A. Students: Gender and previous knowledge

Throughout the different experiences, differences have been detected in the results according to specific characteristics of the students: their gender and previous programming knowledge.

In spite of the positive results obtained, great differences have been detected in the results according to the gender of the students. These differences lay out the need of extending the study before continuing with new implementations.

Due to these differences in the results, the developed studies should be extended to analyse, for example, whether the problems are caused by the selected environments or by the chosen exercise types.

The previous programming knowledge of the students is a latent problem that must be adequately treated. In the answers to the surveys, great differences regarding the motivation and acceptance of the tools were detected according to the previous knowledge of the students.

A way to answer to this issue can be to use different programming environments for each kind of student. As no programming environment is found to be the most adequate to any situation [3], the selected environment could change as the students' progress in their learning process. For the case of LEGO robots, simple graphical visual environment such as Enchanting (http://enchanting.robotclub.ab.ca) or NXT-G could be used with students with no previous knowledge, whereas students with previous knowledge could directly begin with specialised libraries such as LeJOS (http://www.lejos.org).

B. Selection of supporting tools

On the one hand, some difficulties inherent to the use of physical devices have been detected, but these devices should not be automatically rejected as greatest motivation has been detected including physical devices compared to the use of virtual environments. However, this implies the need to design the exercises taking contextual aspects into account [11].

On the other hand, visual environments easily allow abstracting from real-world details and focusing on the logic of the application.

Moreover, it seems interesting to combine different tools. Regarding the integration of physical devices and

development environments, in general there are no problems as physical devices include libraries to use their functionalities, whereas visual environments allow extending their functionality through plugins. However, most programming environments lack deployment capabilities, which allow easily transferring student-developed programs to the devices. To this end, the use of a specific compiler is required. LeJOS library provides this functionality for Eclipse in Java programming language.

C. *Integration in the course*

In order to adequately apply Kolb's experiential learning cycle, it is required to promote that students observe, reflect and find answers to the exercises. Visual programming environments have proved to be appropriate for these purposes.

On the other hand, it is also required to better integrate visual programming environments and physical devices in the evaluation process of the course, or either relate them to other programming area courses in order to provide a wider perspective. Therefore, it could also be interesting to use the some tool in IP and then in MOOP.

VI. CONCLUSIONS AND FUTURE LINES

The experiences carried out suggest that an adequate combination of visual environments and educational robots can improve students' motivation in MOOP. However, the great heterogeneity in previous knowledge of first year students can affect the implementation of improvements such as the ones presented in this paper. Therefore, the use of this tool types require properly adjusting the use of the programming environments to the previous knowledge of the students, and also adequately managing the quantity of environments to be used in each semester, especially if they are new for the students.

On the other hand, an adequate integration of new tools in the course requires that they are somehow included in the evaluation system. This way, a greater acceptance and motivation from the students can be obtained. Having a continuous evaluation process is also positive in this sense.

We are actually working in the implementation of a new combination of educational robots and visual programming environments in MOOP.

References

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

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

[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 Robot., vol. 7, n.o 3, pp. 124-131, 2003.

[4] 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.

[5] A. Wilson y D. C. Moffat, «Evaluating Scratch to introduce younger schoolchildren to programming», Proc. 22nd Annu. Psychol. Program. Interest Group Univ. Carlos III Madr. Leganés Spain, 2010.

[6] D. O'Sullivan y T. Igoe, Physical Computing: Sensing and Controlling the Physical World with Computers, 1st edition. Thomson, 2004.

[7] M. Sartatzemi, V. Dagdilelis, and K. Kagani, «Teaching Introductory Programming Concepts with Lego MindStorms in Greek High Schools: A Two-Year Experience», en Service Robot Applications, InTech, 2008.

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

[9] S. Georgantaki y S. Retalis, «Using educational tools for teaching object oriented design and programming», J. Inf. Technol. Impact, vol. 7, n.o 2, pp. 111–130, 2007.

[10] A. Álvarez and M. Larrañaga, «Experiences Incorporating Lego Mindstorms Robots in the Basic Programming Syllabus: Lessons Learned», J. Intell. Robot. Syst., vol. 81, n.o 1, pp. 117-129, January. 2016.

[11] A. Alvarez and M. Larrañaga, «Using LEGO Mindstorms to Engage Students on Algorithm Design», en Frontiers in Education (FIE), 2013, pp. 1346-1351.

[12] B. S. Fagin y L. Merkle, «Quantitative Analysis of the Effects of Robots on Introductory Computer Science Education», J. Educ. Resour. Comput. JERIC, vol. 2, n.o 4, dic. 2002.

[13] J. Maloney, M. Resnick, N. Rusk, B. Silverman, y E. Eastmond, «The Scratch Programming Language and Environment», ACM Trans. Comput. Educ., vol. 10, n.o 4, pp. 1-15, nov. 2010.

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

[15] M. Kölling, «The Greenfoot Programming Environment», ACM Trans. Comput. Educ., vol. 10, n.o 4, pp. 1-21, nov. 2010.

[16] K. Orton-Johnson, «'I've stuck to the path I'm afraid': exploring student non-use of blended learning», Br. J. Educ. Technol., vol. 40, n.o 5, pp. 837–847, 2009.