Using Alice Software with the Expository Method: A Pilot Study

Joana Martinho Costa [#1]
[#] Institute of Education, University of Lisbon
Alameda da Universidade, 1649-013 Lisbon, Portugal
[1] joana.martinho.costa@campus.ul.pt

***Abstract*—This article presents a learning environment design that compares the effectiveness of using Alice software with conventional programming languages in the scope of initial programming learning. The model proposes a quasi-experimental design in three stages: i) a pre-test to assess the development of logical reasoning in experimental (n=11) and control (n=11) groups; ii) learning programming with the Alice software and pseudocode or exclusively with pseudocode, in the experimental and control group respectively, both using a expository teaching methodology; iii) applying a validated programming test to assess student knowledge, mediated by logical reasoning in both groups. The pilot study indicated no significant differences in the programming test between experimental and control groups ($p > 0.05$). These results indicate that the effectiveness of the Alice software in programming knowledge might not work with the expository method.**

Keyword-Alice software, learning environment, programming learning, pseudocode, teaching method

I. INTRODUCTION

Teaching programming has been studied in the last decades in search for solutions that mitigate the difficulties in the student learning process. Research indicates that the main constraints to learning are the student lack of interest and mathematical knowledge, the syntactical complexity of first programming language that is taught, and the teaching methods, not always the most appropriate [1] [2]. In order to overcome these difficulties, one solution suggested by the research focuses on the use of microworlds [3]: programming environments that allow the user to move virtual objects using commands [4]. In this context, the Alice software stands out: a three-dimensional microworld created specifically to develop computational and logical reasoning and learn the fundamental principles of programming [5].

Although many studies use Alice software and present promising results, a meta-analytical study conducted by Costa and Miranda [6] concluded that the research designs and the instruments used vary widely among studies, which weakens the analysis of the results. The authors also question whether results might be related with the teaching methodologies used, as none of the studies focused on this variable.

Hattie's [7] work states that teaching strategies have an impact with an effect size of 0.60, but audio-visual or visual teaching methods only have an effect size of 0.22. Additionally, Garlick and Cankaya [8] used Alice in their experimental work and obtained no significant changes in the experimental group. These authors also did not isolate teaching methodology, which leads to the question of whether there is a relation between the academic success, the teaching methodology and the chosen programming language to teach initial programming.

In this article, we propose the creation and validation of an initial programming learning environment designed to be implemented in a Portuguese high school, but which can be adapted to different means and countries with a similar curriculum in programming. The implementation of this learning environment can further our understanding of whether the use of the Alice software with pseudocode is more effective in initial programming learning than an approach that exclusively uses pseudocode, and whether the development of logical reasoning influences this relation. The use of pseudocode as an initial approach mirrors the programming student curriculum, on which the experimental work will focus [9]. This design should also allow us to understand whether positive development of logical reasoning is related with the best programming results, regardless of the programming language.

When operational, for a given teaching method and stage of logical reasoning, students using the Alice software with pseudocode to learn the structural concepts of programming will perform better in the programming knowledge test than students who only used pseudocode to learn the same concepts.

After the experimental work, one need understand the evidence about the effectiveness of the Alice software in initial programming learning while controlling the teaching methodology. This article follows Creswell's proposal [10] for the presentation of a research design with a quantitative methodology.

II. METHODS

The next sections describe the methodology, discriminating the operationalization of variables, the sampling strategy and the adopted procedures.

A. Operational variables

This section presents the observable criteria and measurement of variables. The clearness of the objectives and the role assigned to each variable are important issues for the reader to understand what was done and to faithfully replicate this design [11].

Initially, the experimental and control groups were divided using Alice with pseudocode or exclusively pseudocode. This separation corresponds to levels of the study's independent discrete variable: the Programming Language.

The effects of Programming Language were measured with a previously validated programming test [12], and using the logical reasoning test applied before the experimental treatment as a co-variable (ECDL in Table 1). Changes in programming learning were evaluated by estimating the interaction between Programming Language and Performance in Programming Test, mediated by the result of the logical reasoning test.

The equivalence between groups was controlled by testing, before the experimental treatment, whether groups belonged to the same professional course and age range, and whether they had similar prior programming interest, similar grades in Portuguese and Mathematics in the previous year, and similar number of grade retentions. The same teaching method was assured in both groups participating in the experiment, to ensure it would not be a bias factor.

Table 1 presents a summary of all variables that were controlled and manipulated, including the type and operational definition of each variable based on the observable criteria.

TABLE I. Operational Variables

Name	**Type**	**Operational Definition**
Programming Language	Independent discrete	Manipulation: one group learns the structural concepts using pseudocode and Alice approach (level 1), and another group learns the same concepts, but exclusively using pseudocode (level 2)
ECDL	Co-variable continuous	Dynamic property: test score before the experimental treatment
Performance	Dependent continuous	Dynamic property: test score of each student, applied after the experimental treatment
Professional Course	Control	Static property: Students belonging to the same grade and course
Age	Control	Static property: Students belonging to the same age range
ProgrammingInterest	Control	Static property: Students with similar programming interest
PreviousProgrammingExperience	Control	Static property: Students with the same previous programming experience
Schooling	Control	Static property: Students with similar previous schooling regarding the Mathematics andPortuguese grades and to the number of retention years
Teaching Method	Control	Static property: Using the same teaching method in the experimental and control groups during the experimental treatment period
ProgrammingLearning	Intervenient	Inferred from the programming test performance mediated by the ECDL result

B. Instrumentation

The developmental stage of each student's logical reasoning was evaluated before the experimental treatment using the Echelle Collective de DéveloppementLogique (ECDL). This scale classifies the student thinking according to the five stages of Jean Piaget's theory on cognitive thinking, namely: Preoperational, Concrete Operational, Intermediate, Formal A and Formal B (cf. [13]).

In order to evaluate acquired knowledge of programming, we used a test on pseudocode developed and validated by Costa and Miranda [12]. This test contains 13 items, 10 of which are multiple choice and three are development items. Each answer was classified with zero if incorrect, or one point if correct. In the development items, the correct answers were graded using a checklist of instructions needed for the correct solution. If an item on the checklist was completed, the student received that point.

Interest in programming was verified through a survey with clear statements, where students there level of agreement with the statement, on a Likert scale between one and five, where one means they completely disagreed and five means they completely agreed. The statements presented in the survey were: i) "I have an interest in the content of the programming subject"; and ii) "In the future I want to program in my profession".

C. Subjects

The students participating in the pilot experiment all belonged to the same school and class. In the beginning of the school year, and before the experiment, the class was divided by the school in two groups. Both the control and experimental groups included 11 students, with an average age of 15.58 years (standard deviation of 1.31) and 15.67 years (standard deviation of 0.98), respectively. None of the subjects had previous experience in programming. Both the control and experimental groups had four students who had already been retained in previous years.

This class belonged to the 10th grade, the first year of an Informatics professional course. In Portugal, evaluations until the 9th grade in the different subjects are graded between one and five, with one and two indicating negative grades and three, four and five values indicating positive grades. In Mathematics, the control group had an average grade of 2.75 and the experimental group an average grade of 2.67. In Portuguese, the control group had an average grade of 3.1 and the experimental group an average grade of 3.

Interest in programming was verified in the last class before beginning the experimental treatment. The survey was applied to each student in both groups. The control group totalled on average 4.41 (standard deviation of 0.51) and the experimental group answered on average 4.5 (standard deviation of 0.52).

These data allowed confirmed the experimental and control groups were similar in professional course, schooling, age, and previous programming interest.

III. PROCEDURES

We propose a quasi-experimental design, as the population is composed of students previously grouped by classes. Tuckman [11] refers that a certain randomness regarding the constitution of classes by the schools is expected. However, the author also states that research cannot consider this process and should assume it is not possible to divide students according to a predefined criteria.

Based on the characteristics of a quasi-experimental design, the following general procedures were defined: i) data collection about participant characteristics, and application of the ECDL; ii) experimental treatment during four weeks, with seven hours of programming per week, in both groups; and iii) after the experimental treatment, application of the knowledge test in both groups.

$$\begin{array}{ccc} O_1 & X & O_2 \\ \hline O_3 & X & O_4 \end{array}$$

Fig. 1. Visual modelofthe design

D. Classroom Procedures

Since this research occurred in Portugal, the most common teaching method in Portuguese schools was applied: expository teaching [17]. Classes began with an explanation by the teacher about the subject's content, followed by the presentation on an example and respective resolution. Then, the teacher would present students with some exercises to solve, allowing some time for this activity, after which the teacher would solve those exercises on the board if they were in pseudocode or would project the computer screen, if they had been solved using the Alice software. The difficulty level of the exercises gradually increased throught classes. Most of the exercises were solved by the teacher, but sometimes the class was invited to participate, without addressing a specific student. This method was used in both groups.

Table 2 represents the weekly activities during the experimental period. During the first week, both groups learned language syntax in pseudocode. In this week, a researcher was present in the classroom with both groups in order to faithfully replicate the teaching method in the experimental group and to guarantee students would not being able to identify whether they were in the experimental or control group.

TABLE II. Chronogram of the Activities

Week	**Experimental Group Activity**	**Control Group Activity**
One	Pseudocode syntax, survey regarding the programming interest and application of the ECDL	
Two	Syntax and algorithmic notions, first Alice programs and conversion to pseudo-code	Syntax and algorithmic notions, first programs in pseudocode
Three	Programming classes with Alice and conversion of the programs to pseudocode	Programming classes exclu-sively in pseudocode
Four	Programming classes with Alice and conversion of the programs to pseudocode	Programming classes exclu-sively in pseudocode
Five	Programming classes with Alice and conversion of the programs to pseudocode	Programming classes exclu-sively in pseudocode
Six	Application of the knowledge test	
Seven	Programming language C syntax learning	

The curriculum was followed in the same way, addressing the same concepts in both groups that participated in the research. In the first week, students learned the pseudocode syntax and the initial data were collected. In the four following weeks, the classes of the control group were guided by the subject's teacher, and the following structural programming concepts were discussed: algorithm development, variables, data types, operators and control structures. During these four weeks, some classes of the experimental group were guided by the researcher and others were guided by the subject's teacher, who also guided the control group. The subject's teacher was present in all classes. In this way, the teaching method was assured in both groups and it was possible to control the teacher and the teaching method variables. t

In the fifth week, after completion of the experimental treatment, both groups were submitted to the programming knowledge test [12]. After that, all the students started learning the C language, as taught by the subject's teacher.

E. Validity Procedures

In order to ensure the validation of the research and avoid any possible distortion caused by the selection, the ECDL was applied as a pre-test to evaluate the existence of initial equivalence regarding logical reasoning ability. A possible distortion of the selection was also overcome by including the following control variables in the analysis: age, course, previous programming experience and programming interest. Although the use of a pre-test constitutes a threat to validation in testing [11], it was the only possible way to analyse the results of the programming knowledge test, using the logical reasoning development as co-variable. The knowledge test had no pre-test and therefore its application did not constitute a threat to the validity of the re-search. During the research period, groups were submitted to the same experiments in order to control the effects of maturation, history and statistical regression [11].

According to Tuckman [11], the Hawthorne effect, that is, the set of effects caused by subject motivation for merely participating in an experiment, constitutes a threat to the external validation of research. To control this effect, we followed Tuckman's [11] proposal: the control group of the experiment, which in traditional research would only contact a researcher during data collection before and after experimental treatment, was transformed into a Hawthorne Group. This way, the control group had constant interactions with a researcher, including their presence in classes during the experimental treatment. During this time, questions irrelevant to the experiment were asked in both groups. These questions had no effect related to the experimental treatment,but made it possible to keep the same motivation levels in both groups.

The validation of the instrumentation was previously verified using valid and reliable instruments.

IV. Ethical Issues

Every ethical issue regarding the research were ensured (cf. [15]). Participation in the research was authorized by the school and students themselves. The students who were 17 years old or younger were requested to provide an informed consent from their tutors, in order to participate in the research. All participants were also informed that their data would be protected by anonymization. All data collected was used exclusively for this research.

The information provided to participants, school and tutors stated that students would not suffer any mental or physical danger from the procedures used in the research and that any relevant change in procedures would be previously transmitted. Participants were free to not authorize the inclusion of their data in the study, even after their participation, and that they could leave the research in any phase. The informed consent included the right to voluntarily participate, the purpose of the study, procedures, the right to their privacy, to ask questions and the right to access the results.

The students who were part of the class, but who had previous experience in programming, were not considered for the results, but also participated in the research without knowing that their data would not be used, so that they would not feel excluded.

After a few weeks, some classes were taught with the Alice software in the control group to ensure the equality of opportunities in access to the knowledge about this microworld.

V. Preliminary Results

F. Assumptions

The effect of the programming language on the programming knowledge test was evaluated with ANCOVA using the ECDL results before the experimental treatment as a co-variable. The ANCOVA assumptions were validated through the Shapiro-Wilk test of normality of the dependent variable ($p \geqslant 0.142$ to both groups). This test is preferable to the Kolmogorov-Smirnov test, because the number of subjects in the sample was smaller than 30 [16].

The homogeneity of variances was verified using the Levene test ($p = 0.437$). Homogeneity of slopes was validated through the procedure described in Marôco [16]: verification of the significance of the interaction between the co-variable and the factor, in the two-way ANOVA ($p = 0{,}328$).

Homogeneity of the means of the co-variable was also evaluated in the two-factor levels ($p = 0.076$) [16]. All statistical analysis was performed with SPSS Statistics (v. 24, IBM, SPSS, Chicago), with a significance level of 0.05.

G. Pilot Findings

The results of the programming knowledge test were not significantly influenced by the Alice software, after accounting the ECDL as a co-variable ($F(1,19) = 0.984$; $p = 0.334$).

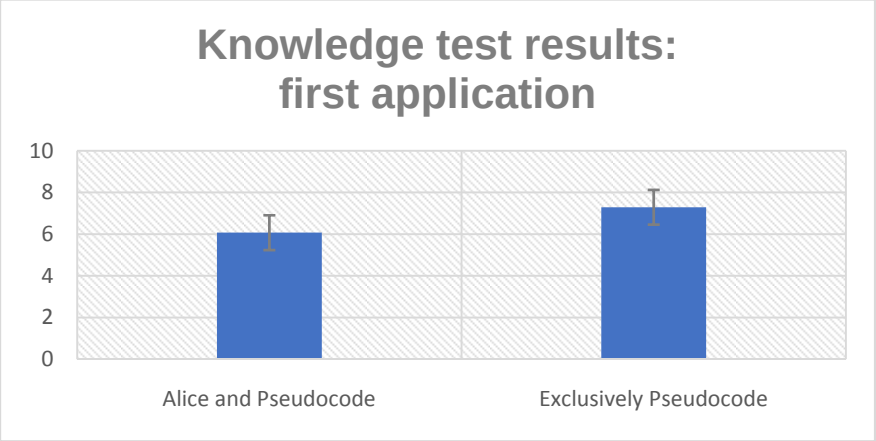

Fig. 2. Average in both groups for the same value of the ECDL pre-test (first application)

As shown in Figure 2, when the ECDL co-variable is maintained at its mean value (11.95), the group that used Alice scored lower in the programming test (6.073 ± 0.834) than the group that exclusively used pseudocode ($7{,}290 \pm 0{,}834$). The analysis of planned contrasts between the experimental and control groups revealed non-significant differences between the approaches ($\hat{\psi}1 = 1.217$; $t(19) = 0.992$; $p > 0.05$).

VI. CONCLUSIONS AND FUTURE WORK

Stefik and Hanenberg [17] question whether most studies have enough rigor and impartiality to analyse the impact of the newest emerging solutions, as part of these studies are done by the teams that created these solutions. Moreover, Rosenberg [18] points out that there is a clear trend towards the publication of statistically significant results, which can create a favourable bias regarding the real impact of these solutions. The results obtained in our pilot-group focus on these issues and on the need to maintain research over a longer period and include variables often lacking in other studies.

The choice of quantitative methods was adequate for the proposed research questions, as only quantitative data would allow the analysis of the programming test results. The sample was also appropriate for constituting a pilot group for this project, given the characteristics of the subjects. However, the design could improve by including a larger number of students and a comparison with a group using the Alice software, but with a different teaching methodology. This shall be implemented in the future to complete this research.

These results suggest that the use of the Alice software with an expository teaching method does not bring any benefit to programming knowledge when compared with traditional tools. However, this work needs further research: increasing the time of the experiment, using other teaching methods associated with the Alice software and other programming languages, and using the same instruments to get valid conclusions. Following this approach, it should be possible to verify wheter the use of microworlds in programming learning is more effective in acquiring programming knowledge than the use of conventional programming languages, or whether this acquisition is more related with the teaching method used than with the teaching tools, or whether it will be the combination of both. It Possible, the absence of significant results in the present, unlike the intentions of the creators of the Alice software, might mean that most programming languages that currently exist have the same potential in initial programming learning.

REFERENCES

[1] Qian, Y., & Lehman, J. (2017). Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review. Transactions on Computing Education, 18. doi: https://doi.org/10.1145/3077618
[2] Gomes, A., Henriques, J., & Mendes, A. (2008). Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. Educação, Formação&Tecnologias, 1, 93-103.
[3] Moons, J., & Backer, C. (2013). The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism. Computers and Education, 60, 1, 368-384. doi: https://doi.org/10.1016/j.compedu.2012.08.009
[4] Papert, S. (1985). Different visions of Logo. Computers in the Schools: Interdisciplinary Journal of Practice, Theory and Applied Research, 2, 3, 3-8.
[5] Alice Project. (2018). Alice – Tell Stories. Build Games. Learn to Program. Retrieved from http://www.alice.org.
[6] Costa, J. M., & Miranda, G. L. (2017). Relation between Alice software and programming learning: A systematic review of the literature and meta-analysis. British Journal of Educational Technology, 48, 6, 1464-1474. doi: http://dx.doi.org/10.1111/bjet.12496
[7] Hattie, J. (2015). Visible Learning into Action (1st. ed.). New York: Routledge.
[8] Garlick, R., &Cankaya, E. (2010). Using alice in CS1: a quantitative experiment. Proceedings of the 15th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (pp. 165-168). doi: https://doi.org/10.1145/1822090.1822138
[9] Direção-Geral de Formação Vocacional (2005). Disciplina de Programação e Sistemas de Informação: Programa Componente de Formação Técnica. Ministério da Educação, Lisboa.
[10] Creswell, J. (2003). Research Design: Qualitative, quantitative, and mixed method approaches (2nd ed.). California: Sage Publications.
[11] Tuckman, B. (2012). Manual de Investigação em Educação: Metodologia para conceber e realizar o processo de investigação científica (4th ed.). Lisboa: Gulbenkian.
[12] Costa, J. M., & Miranda, G. L. (2017). Desenvolvimento e validação de uma prova de avaliação das competências iniciais de programação. RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação, 25, 66-81. doi: https://doi.org/10.17013/risti.25.66-81
[13] Hornemann, J. (1974). Influenceducontenusur la résolution de problèmeslogiques. Enfance, 27, 45-64.
[14] Conselho Nacional de Educação (2017). Estado da Educação 2016. Retrieved from http://www.cnedu.pt/content/edicoes/estado_da_educacao/CNE-EE2016_web_final.pdf.
[15] Johnson, B., & Christensen, L. (2004). Educational Research: quantitative, qualitative, and mixed approaches (2nd ed.). Boston: Pearson Education.
[16] Marôco, J. (2014). Análise Estatística com o SPSS Statistics (6th ed.). Pêro Pinheiro: Report Number.
[17] Stefik, A., &Hanenberg, S. (2014). The Programming Language Wars. Proceedings of the 2014 International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (pp.283-299). Portland: ACM. doi: https://doi.org/10.1145/2661136.2661156
[18] Rosenberg, M. (2005). The File-Drawer Problem Revisited: A General Weighted Method for Calculating Fail-Safe Numbers in Meta-Analysis. Evolution, 59, 2, 464-468.

AUTHOR PROFILE

Joana M. Costaholds a Bachelor in Computer Engineering, a Master in Computer Education andshe is a final year PhD student in ICT and Education at the Institute of Education, Universityof Lisbon. Her research interests focus on programming learning, microworlds andeducational technology.