# A coding curriculum for K-12 education: the evidence-based approach

**F Vico[1,4], M Molina[1], D Orden[2], J. Ortiz[3], R Garcia[4], J Masa[4]**

[1] *D. Lenguajes y Ciencias de la Computación University of Malaga (SPAIN)*
[2] *D. Física y Matemáticas University of Alcalá (SPAIN)*
[3] *CEIP Intelhorce (SPAIN)*
[4] *ToolboX Academy (Madrid, SPAIN)*

## Abstract

In recent years, public and private educational systems are making efforts to update their programmes as to integrate computational thinking and computer programming in K-12 grades. Despite the expertise of the academic world in teaching CS, and the vast amount of knowledge and tools available to evaluate programming skills in industry, there are no clear clues on how children do learn CS concepts, or what is the best strategy to develop coding skills in the school, and how to implement it with limited resources (hardware and trained personnel). The reason for these difficulties has to do with the variety of environments in the market, as well as the complexity to perform controlled multicenter experimentation with children that are exposed differently to computer programming (from no contact at all, to gaining skills at home, or in academies). We propose a curriculum to teach the most relevant concepts of CS, ranging from the very basic command execution, to problem-solving with Artificial Intelligence heuristics, covering the whole K-12 school grades. Instead of being based on computational thinking studies, this curriculum is the result of four years of field work, testing the ToolboX Academy programming environment on site, from a direct scrutiny of children interaction with the interface and tasks definition, to a controlled experiment where more than a thousand students became involved, from 30 different centers. The result is a detailed description of how the fundamental concepts of computer programming can be presented to primary-school students in order to reach a high level of coding proficiency, as well as acquiring the AI bases.

Keywords: coding curriculum, artificial intelligence, K-12.

Preview of the version published in Proceedings of the EDULEARN19, 7102-7106, 2019.
ISBN: 978-84-09-12031-4. ISSN: 2340-1117

## 1 INTRODUCTION

There is an emerging consensus about the important role of computer science in today's society. Several authors [1] agree that citizens should aim to have the potential of creating computational tools, not just being mere users of tools created by others. A number of prominent institutions have contributed to the debate about the introduction of computational thinking in compulsory education[2], stating as main reasons promotion of the necessary skills for a digital work, development of logical thinking and problem-solving skills, and boost of the economic growth.

As a result of this interest, many countries have reformed their curricula in order to include computational thinking. In Europe, most of them have done so, with differences between regions arising at countries where curricula depends on regional authorities [2]. For illustration purposes, the K-12 curriculum in England [3] ranges from understanding what algorithms are or use logical reasoning to predict the behaviour of simple programs, to understand key algorithms like sorting and searching or undertake creative projects to achieve challenging goals. In countries like Poland [4] the aims include understanding and analysis of problems or programing and problem solving by using computers and other digital devices, while goals in Croatia [5] range from following simple steps to create digital content to plan, create, present, and evaluate a multimedia project.

Other countries, like Portugal [6], propose specific goals like developing notions related to proportionality, stimulate three-dimensional perception, or lead to the discover of physical notions in an intuitive way. Finally, in the US the K–12 Computer Science Framework [7] provides examples of activities and states that by the end of Grade 12, students should be able to identify complex, interdisciplinary, problems that can be solved computationally, decompose complex real-world problems into manageable subproblems that could integrate existing solutions or procedures, or create a computational artifact.

The examples above illustrate, on the one hand, the lack of consensus on the definition of computational thinking [8] and, on the other hand, the difficulty of proposing concrete contents which respect teachers' freedom to suit to their specific school context. Experts recommend that the computing curricula should provide clear examples [9], being this the main contribution of this paper. The current work stems from an extensive on-field experience of four years of computational activities with a huge variety of 30 schools and more than 1000 students, figures significantly larger than those in the state of the art [10].

Based on such experience, this paper proposes a curriculum to teach the most relevant concepts, ranging from the very basic command execution, to problem-solving with Artificial Intelligence heuristics, covering the whole K-12 school grades. A detailed description of how the fundamental concepts of computer programming can be presented to primary-school students is provided, showing that this allows to reach a high level of coding proficiency, as well as acquiring the AI bases in secondary education.

## 2 METHODOLOGY

In order to explore children's capacity to learn computational concepts, a controlled experiment was designed and performed in cooperation with the Educational Innovation Service of the Regional Ministry of Education and Sports of Junta de Andalucía (the regional government in Andalusia[1]). In this experiment, the students were trained with ToolboX.Academy [11] during one hour of coding practice (an implementation of the Hour of Code concept), where they were asked to solve concrete problems, depending on their age (i.e., the grade they were enlisted in) and without any help from the teacher or other students. The learning method used is by analogy, that is, the student was presented with a problem, and had to interpret the solution (program) that was given, or to copy or correct a proposed solution, or to write a solution by herself. The procedure, data processing, analysis and conclusions for the design of K-12 coding curriculum are described in the following sections.

This study used an exploratory data analysis design where no initial hypotheses were stated. Data were collected from 30 centers where we sought to analyze the difference in coding between the groups. For the selection of participants, the Innovation Service of the Education Counseling made a call to participate in the experiment among the public education centers of Andalusia, with the result of 30 centers recruited. This enrolled 35 educators, with 71 student groups to be involved in the experiment: a final sample of 1055 students, in the range from 6 to 18 years old, covering all K-12 levels. All participating educators, in charge of monitoring the groups of students, were provided with very simple instructions to set up the programming environment and facilitate the individual log in of the students. The instructions were the following: (1) each student must perform the tasks individually; (2) the educator cannot help the student to solve the tasks; (3) each student must enter their unique identification code; and (4) the experiment will last, as much, for 60 minutes.

Some of the students had previously coded with environments like Scratch, but they did not have any previous experience with ToolboX.Academy. All groups, regardless of age, had the same concepts to learn. The only difference between groups of younger and older students was that, for younger ages more tasks were needed in order to understand the concept. This, of course, affects the number of tasks that can be solved (hence, the concepts that are learned) in a fixed time, despite the fact that older students adapted faster to the environment and mechanics of the tool.

1076 students participated in the study and all the anonymised metadata obtained while performing the experiment was received in a server. Mainly due to logistic reasons, some of the students were able to perform the experiment for a very limited period of time, so this

[1] Andalusia is the southern Spanish region, with a pre-college population of around 1 million students in public centers.

initial sample was filtered as to keep the data of those students that were trained for at least 15 minutes, resulting in 1055 subjects (a higher experimentation time could have been imposed, but there was also the case of gifted subjects that would had been filtered out because they simply took very little overall time to solve the tasks). The final distribution of the sample by grades is given in Table 1.

**Table 1. Distribution of the participants by grade.**

| | Primary school | | | | | | Secondary school | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| grade | p1 | p2 | p3 | p4 | p5 | p6 | s1 | s2 | s3 | s4 | b1 | b2 |
| # students | 39 | 43 | 50 | 91 | 49 | 158 | 91 | 167 | 186 | 82 | 74 | 25 |

## 3 RESULTS AND CONCLUSIONS

All participating groups in each grade performed the experiment as to learn the following list of computational concepts, ranked from the very simplest single command program, to the definition of functions, as to produce structured code.

1. single command
2. sequence of commands
3. repeat loop (single-command body)
4. repeat loop (multiple-commands body)
5. sequence of repeat loops
6. input/output
7. variables
8. if-else
9. if-elseif
10. do-until loop with relational operators
11. do-until loop with logical operators
12. while loop
13. general-purpose functions (mathematical libraries)
14. user-defined functions

Fig. 1 shows how the students performed as a function of grade. We can see, for example, that 75% of the students in first grade of primary education were able to complete all the single-command tasks, while 25% completed also the multiple-command tasks, and less than 10% completed the repeat-loop tasks (something remarkable, considering that many of them had not learnt to read, and performed the experiment by themselves, with no external help). As expected, for higher grades, the performance increases. We can see, for example, that half of the students in the last grade of primary education completed the repeat loop block, and even some of them were able to complete the tasks with conditional structures. As for the last grades of secondary education, some students were able to complete the whole set of tasks, demonstrating competence in all the computational concepts under study.

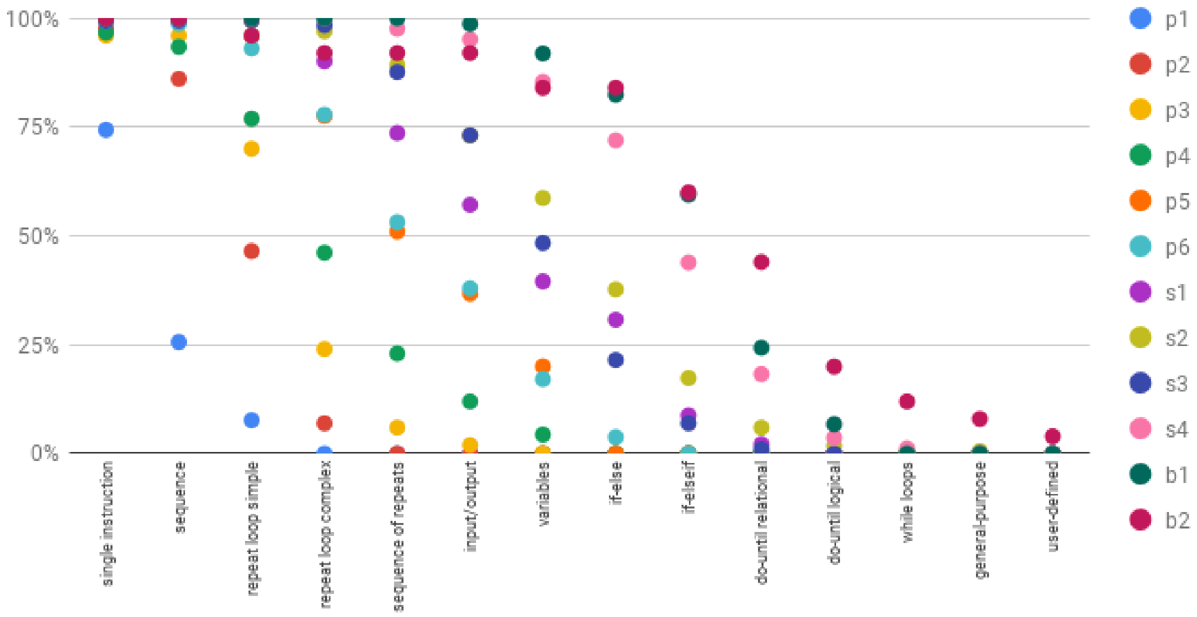


*Figure 1. Computational concepts learned as a function of grade, represented as the percentage of students in that grade that completed all the tasks to master each concept.*

When focusing on the number of solved tasks (a quantitative variable, as opposed to the computational concepts learned), we can compare the overall performance depending on the grade. Fig. 2 shows that the first three grades of primary education have a similar performance on average, completing some 25 tasks in about 40 minutes (dashed line). Interestingly, the 4th grade increases this performance in around 25%, and a similar increment is obtained by the 5th grade, and again for the 6th grade, which can be included in a single cluster of performance grouping all the grades in secondary education (as seen by the tasks solved after 30-minutes -dotted line).

The reasons for this particular distribution of performance are not clear, as this was not the target of the study, but they could be related to the ability to read, and cognitive development (mainly, abstraction capacity).

Certainly, an hour of code is but a way to approach and explore the capabilities of K-12 students to learn coding. While a formal and reliable curriculum can only be based on years of experience teaching children the main computational concepts at the school, the lack of this knowledge forces national and regional governments to make proposals based on simple speculations about their capacity to acquire these skills, which typically results in conservative proposals and efforts.

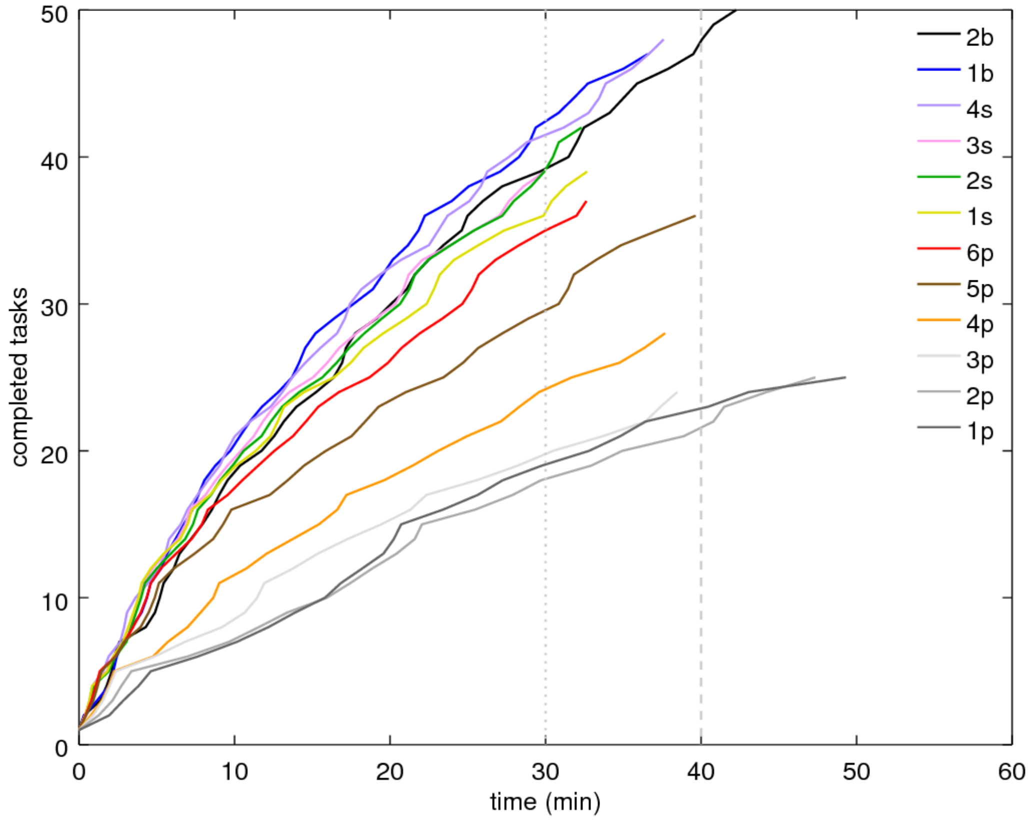


*Figure 2. Average performance curves (measured as the number of tasks solved) for each grade. The first three grades of primary education show similar performance, and it increases at a 25% pace in the next three grades, in a way that the last grade joins the cluster of secondary education grades. Vertical lines are drawn at 30 and 40 minutes of experiment duration, so the grades can be compared.*

The results of this controlled experiment, and the experience in the classroom with the second cycle of preschool (3-5 years-old children, helped by primary education students) and with Artificial Intelligence tasks in secondary education, suggest three important facts: (1) that students can start coding at a very early age in environments that have been specially designed and fine-tuned to their capacities; (2) that students can complete their education in coding by the end of the primary education; and (3) that, having acquired these skills in primary education, students of secondary education can extend their knowledge to more abstract and elaborated concepts, like object-oriented programming and AI-based algorithms.

ToolboX.Academy bases its approach in this evidence, providing a pre-exposure to computational thinking to preschool students, a complete coding curriculum for primary education, and an advanced-programming and AI-based curriculum for secondary education.

Evidence suggests that this proposal is far from being bold, instead it seems to be realistic (considering the degree of involvement that students show in repeated sessions of coding learning), and touches one of the main problems that our society has been posed in the advent of the industrial and cultural revolution that AI is paving, and the inequality in the job market that it will surely bring about.

**ACKNOWLEDGEMENTS**

This study would hardly have been performed without the help, enthusiasm and involvement of the Educational Innovation Service of the Regional Ministry of Education and Sports of Junta de Andalucía. The authors are indebted to the educational centers (particularly to Intelhorce and La Biznaga primary schools, and Emilio Prados high school, for helping in the initial steps of this experimentation phase), and to the Department of Computer Science and Artificial Intelligence for providing the server to store usage data.

**REFERENCES**

[1] B. Czerkawski. Computational thinking in virtual learning environments, in *Proceedings of E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, 2015, 993-997, 2015. Chesapeake, VA: Association for the Advancement of Computing in Education (AACE).

[2] S. Bocconi, A. Chioccariello, G. Dettori, A. Ferrari, and K. Engelhardt. Developing computational thinking in compulsory education-Implications for policy and practice, (No. JRC104188), 2016. Joint Research Centre (Seville site).

[3] Department of Education. National curriculum in England: computing programmes of study, 2013. Retrieved from https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study on May 22, 2019.

[4] M.M. Sysło and A. B. Kwiatkowska. Introducing a new computer science curriculum for all school levels in Poland, in *International conference on informatics in Schools: Situation, evolution, and perspectives*, 2015, 141-154. Springer.

[5] L. Kralj. New Informatics curriculum—Croatian tradition with world trends, in *39th International Convention on Information and Communication Technology, Electronics and Microelectronics* (MIPRO) 2016, 760-763. IEEE.

[6] Direção-Geral da Educação. Linhas Orientadoras para a Robótica, 2016. Retrieved from http://www.erte.dge.mec.pt/sites/default/files/linhas_orientadoras_para_a_robotica.pdf on May 22, 2019.

[7] K–12 Computer Science Framework, 2016. Retrieved from http://www.k12cs.org on May 22, 2019.

[8] V. J. Shute, C. Sun, and J. Asbell-Clarke. Demystifying computational thinking, *Educational Research Review*, **22**;142-158, 2017.

[9] M. Webb, N. Davis, T. Bell, Y. J. Katz, N. Reynolds, D.P. Chambers and M.M. Sysło. Computer science in K-12 school curricula of the 2lst century: Why, what and when?, *Education and Information Technologies*, **22**(2), 445-468, 2017.

[10] C. Chalmers. Robotics and computational thinking in primary school, *International Journal of Child-Computer Interaction*, **17**:93-100, 2018.

[11] F. Vico, R. Garcia & J. Masa. ToolboX.Academy: Coding & Artificial Intelligence made easy for kids, Big Data for educators, in *Proceedings of the 11th annual International Conference on Education and New Learning Technologies* (EDULEARN19), 5173-5178. 2019.