Dodero Beardo, J. M., Mota, J. M., & Ruiz-Rube, I. (2017). Bringing computational thinking to teachers' training: a workshop review. In J. M. Dodero, M. S. Ibarra Sáiz, & I. Ruiz Rube (Eds.), Fifth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'17) (Cádiz, Spain, October 18-20, 2017) (pp. Article 4). New York, NY, USA: ACM. doi:10.1145/3144826.3145352

Bringing computational thinking to teachers' training: a workshop review

Juan Manuel Dodero
University of Cadiz
Spain
juanma.dodero@uca.es

José Miguel Mota
University of Cadiz
Spain
josemiguel.mota@uca.es

Iván Ruiz-Rube
University of Cadiz
Spain
ivan.ruiz@uca.es

ABSTRACT

In recent years, several visual programming languages and tools are emerging, which allow young students to easily program applications. Particularly, the block-based language used by Scratch has been the standard in most school initiatives to introduce Computational thinking (CT) in courses unrelated to computing. However, CT competences are not specifically included in the curricula of many Higher Education degrees that future teachers of Primary and Secondary Education have to complete. This paper describes a workshop for teachers' training on CT. It is based on the block-based common language of Scratch, but focused on enhancing teachers' skills to develop mobile applications with a tool based on the MIT's AppInventor. This workshop provided some insights on the capabilities of future teachers in the use of programming tools.

CCS CONCEPTS

• Social and professional topics → Computing education • Human-centered computing → Ubiquitous and mobile computing design and evaluation methods

KEYWORDS

Computational thinking; mobile programming; visual programming languages.

1 INTRODUCTION

Computational thinking (CT) refers to a collection of computational ideas and habits of mind that people in computing disciplines acquire through their work in designing programs, software, simulations, and computations performed by machinery [1]. Although computational thinking is different from computer programming, it includes all factors that are essential to coding. There have been different initiatives, such as code.org [2], as an effort to bring computer science and computational thinking skills to young people, both inside and out of schools. Nonetheless, most frequent approaches to teaching digital literacy have been based on the learning of programming [2].

Having their roots in the 70s and 80s with the introduction of coding for educational uses, mainly through the Logo programming language [3], new visual programming languages such as Alice[3], Kodu[4], Scratch[5] and AppInventor[6] have emerged more recently. These languages allow young students to program applications without the need to learn the complex syntax of the traditional programming languages [4], thus fostering computational thinking skills by overlooking hindrances of traditional computer programming languages.

Many schools are including in their curricula the use of mobile devices, which are provided by the school, while others adopt a bring-your-own-device (BYOD) solution for learning. Anyhow teaching and learning practice in schools is radically changing due to such technologies, so that teachers need to find solutions to professional issues related to the use of a

[2]https://code.org
[3]http://www.alice.org
[4]https://www.kodugamelab.com/
[5]https://scratch.mit.edu/
[6]http://appinventor.mit.edu

technology they have to learn to live with [5]. However, CT competences are not specifically included in the curricula of many Higher Education degrees that future teachers of Primary and Secondary Education have to complete. In the European context, an ample overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers has been developed in the TACCLE 3 project [6,7]. The lack of training in CT is ample also in Spain, where it usually requires to be addressed through self-study and off-school training, particularly in the under-18 education levels [8,9]. This is not different from what happens in the K-12 levels of education in many countries.

Several teacher-oriented workshops have increased teachers' understanding of CT and how to integrate it as part of the curriculum [10,11,12]. In some universities, a CT training program has been developed for future teachers based on Scratch [13]. Actually, the block-based language used by Scratch has been the standard in most school initiatives to introduce CT concepts and practices in courses unrelated to computing [14]. This work poses a slightly different approach to design a workshop for teachers' training on CT. It is based on the block-based common language of Scratch, but focused on enhancing teachers' skills to develop mobile applications with the MIT's AppInventor tool [15].

The rest of this work is structured as follows: Section 2 presents the software tool developed with the aim of easing the inclusion of diverse computing technologies in educational contexts. Section 3 describes the experience conducted for introducing CT concepts to the students of a Master's Degree in Teacher Training. Finally, the conclusions and future work are shown in Section 4.

2 THE VEDILS PLATFORM

2.1 Authoring Tool

This tool was developed with the aim of easing the inclusion of diverse computing technologies in educational contexts. Instead of creating a new specific tool, which may limit the choices for some teachers who might have a relative experience with computer programming, the VEDILS platform is based on MIT's App Inventor, which is an easy-to-use online tool for developing Android mobile apps. Using this tool, users without strong programming skills are able to design and build by themselves mobile apps, thus democratizing mobile programming.

Figure 1: Design view of VEDILS.

This tool provides a simple drag&drop view (Fig. 1) for designing the app's user interface and a programming language based on visual blocks (Blockly) to declare its behavior (Fig. 2).

Figure 2: Blocks view of VEDILS.

Unlike AppInventor, VEDILS provides a set of extensions to develop Augmented Reality (AR) and Virtual Reality (VR) scenarios that use diverse HMI technologies for multimodal interactions, such as gestural and brain interfaces, as well as the capabilities for supporting learning analytics

2.2 Measurement tool

During the development process of mobile apps, some questions may arise: how many blocks are using the users to define the behavior of a simple app? are they using procedures or functions in their programs? which are the components most used in their apps? how long do users need in average to develop the apps? how many builds and debugs perform the users while developing the apps? who are the most active developers? With the aim of providing answers to these and other questions a data analytics platform has been set up. This solution provides a set of data schemes for multidimensional analysis and a web dashboard (Fig. 3). All these components have been designed using the Pentaho BI suite.

Figure 3: VEDILS main measurement dashboard.

3 WORKSHOP

3.1 Settings

A workshop for introducing CT concepts to the students of a Master's Degree in Teacher Training at the University of Cádiz was conducted. Students attending this course belong to different educational areas, which range from Science and Maths to Technology to Social Sciences to Language and Literature to Physical Education and Visual Arts. The Master Degree qualification is required in Spain to teach in secondary schools. Students were introduced to CT by means of the VEDILS authoring tool.

The workshop was held in two 4-hour sessions taking place during two weeks. An overall of 22 students as future teachers of different educational areas and without previous programming knowledge were the participants. During the first session, a series of guided exercises were conducted to learn programming concepts. These exercises are detailed in Table 1, Table 2, Table 3 and Table 4.

Table 1: First exercise: welcome to class

Components	Buttons, Labels, Text to Speech
Target	Enter a name in the text box and then voice playback that text by adding the phrase "Welcome to class" at the beginning
Learning	introduction to events, use of getter and setters, procedure calls, text concatenation
Additional	During the first exercise, the VEDILS Companion tool was introduced to enable online debugging of the application on a mobile device.

Table 2: Second exercise: tell me.

Components	Buttons, Labels, Layout components, SpeechRecognizer, Camera, Share, PhoneCall
Target	The application must recognize voice instructions and, according to the word dictated, response with: taking a picture to send by message; making a phone call; setting a screen background image.
Learning	Use of conditional control blocks, response to events

Table 3: Third exercise: Switching between screens.

Components	Buttons, Labels, Layout components, Screens
Target	Create multiple screens and switch between them using buttons
Learning	Grouping of contents by screens, importance of the initial screen defined by default.

Table 4: Fourth exercise: Story with augmented reality.

Components	Buttons, Labels, Sound, ARCamera, ARMarkerTracker, AR3DModelAsset
Target	Create a story using AR components. Two marks of AR have to be associated with 3D images and another with a 2D image, which will serve as background and associated with a sound
Learning	Grouping of contents by screens, importance of the initial screen defined by default, AR concepts, and how AR can be implemented in educational content to support explanations.

At the end of the session, students were asked to come up with ideas for an application that they had to create within their area of knowledge. The second session began by designing on paper how the screens, their content and the flow between them should be. Each student deepened in the learning of the tools that were more necessary for their application area, always with the teacher's support, to complete their applications before the end of the class.

As a final objective of the course, students had to develop and deliver an application, different from the one created in class, where they would apply the concepts learned. This has been the application evaluated in this paper.

Students had no prior programming skills and were still able to follow the course without major problems. The only issues encountered were due to the mobile devices where the applications were tested, sometimes because of a lack of memory and others due to the Android operating system version. Initially some students did not want to free up space on their devices to install the applications, but when they checked the possibilities the programming environment provided, they did not hesitate to delete pictures and messages to do it. Thus, the lack of planning is a common problem when developing the most complex exercises, because students started to program the applications directly without having designed it previously.

3.2 Results

A review of the apps developed by the students is presented in this section, along with a set of metrics related to the complexity and the variety of the components and programming language instructions used in the apps.

Most applications had the following structure: an initial screen asking for the user's name, which was used on the other screens to customize the application; a presentation of concepts on the educational area where the application was developed; a number of evaluation exercises of the concepts learned (e.g. applications for reinforcing foreign language learning used components such as speech recognition); and a farewell screen where the exercises scores were displayed.

The work exceeded the initial expectations of students, as they did not expect to be able to develop educational content for mobile devices. The future teachers discovered the possibilities offered by developing applications to teach their future students in primary or secondary schools, as well as the possibilities provided by mobile devices that are not being fully exploited as a curricular subject.

Among the works, two of them especially stand out. One of them consisted in learning about genes to continue with a series of questions (see Fig. 4).

The second remarkable application used AR capabilities for learning about perspectives, using AR marks associated with 3D models that the student designed (see Fig. 5).

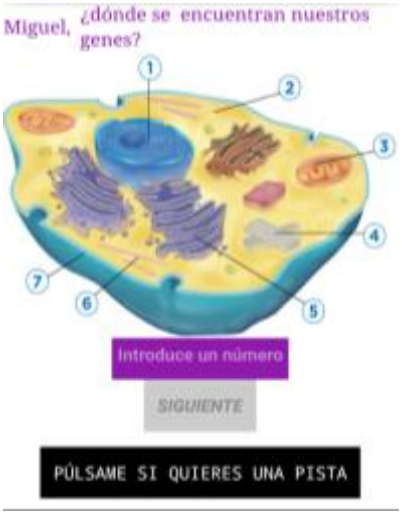

Figure 4: App for learning about genes.

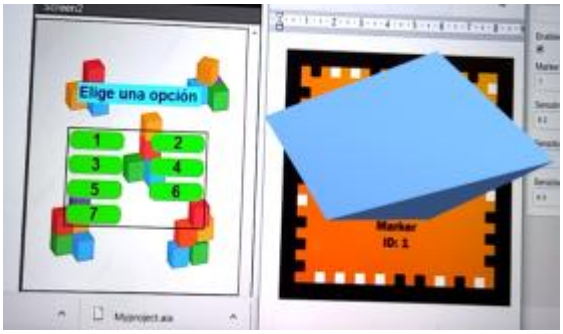

Figure 5: Augmented reality application to learn about perspectives.

Some basic statistics (see Table 5) were computed by using the VEDILS analytics tool, such as the time needed to develop the apps; the number of screens, components and blocks used and the number of build and debug processes launched through the VEDILS environment.

Table 5: Basic statistics of developed apps.

Metric	Average	Std dev
Duration	4:19:30	3:05:53
Screens	7	4.57
Components	63.76	43.83
Blocks	157.76	157.02
Builds	2.86	4.88
Debugs	19.71	16.19

Fig. 6 depicts the diversity of both visual and non-visual components used by students to design the mobile applications. User interface elements, such as labels, buttons, textboxes and images, and layout containers (vertical and horizontal) are mostly used components in the developed

mobile apps. To a lesser extent, multimedia elements, such as the *TextToSpeech* component, are also frequently used.

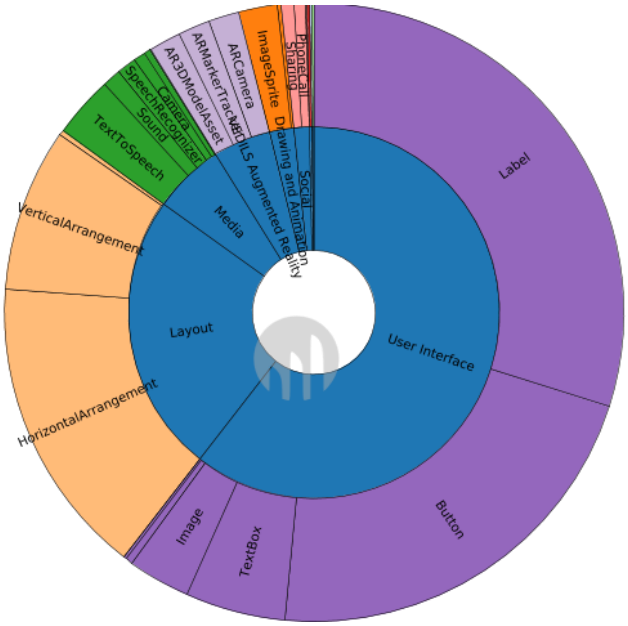

Figure 6: Distribution of the components used in the developed apps.

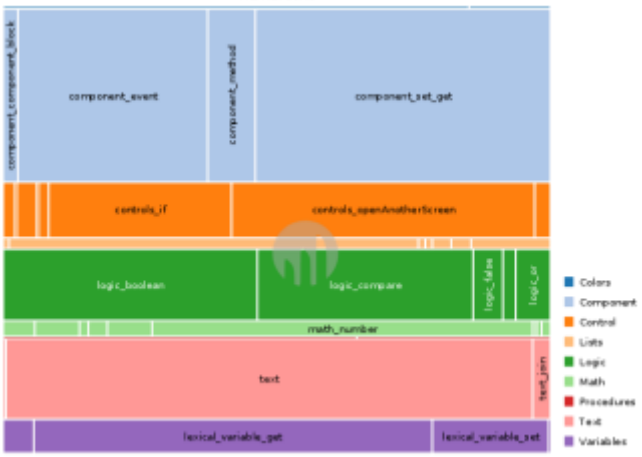

Figure 7: Distribution of number of blocks.

The number of blocks used by students to develop their apps were computed and broken down into the following categories:

- Component: Amount of received events, invoked functions or properties accessed/modified on the app's components.
- Control: Amount of flow control instructions used.
- Lists: Number of lists created and operations applied.
- Logic: Amount of boolean values and logic operations.
- Math: Amount of numerical values and math operations.
- Text: Amount of literal values and string operations.
- Procedures: Amount of definitions and invocations to procedures.
- Variables: Amount of declarations and use of variables.

Fig. 7 depicts the range of blocks used to define the behavior of the apps. Most of those blocks correspond with event handlers and read/write access to components' properties. Then, some control flow blocks, such as the conditional *if* and the *openAnotherScreen*; logic blocks, such as boolean definitions and comparisons; and blocks for defining text literals are also quite often used. It is worthy to mention the scarce use of loop instructions and procedures.

3.3 Discussion

The previous results have provided us with a number of insights about the programming habits and CT principles followed by future teachers as novice developers, namely decomposition, pattern matching, abstraction, and algorithmic principles.

3.3.1 Application appearance. The number of components to arrange items on screen indicates the importance that students give to the appearance of applications. This was one of the topics most asked by the students, as they wanted to give an appearance to their applications similar to the Google's Material Design style, something that was not possible in the mobile app development environment currently used, based on App Inventor. In addition, App Inventor does not allow to copy and paste controls between screens; only code can be copied, so if someone wants the screens to have a common look and feel, they must do it manually for each screen and item.

Another issue frequently encountered is the lack of a previous design of the application. Students were directly to the design and coding in the authoring tool. This caused a lot of changes as the components were inserted, after realizing that it was not the best arrangement. It is therefore important to strengthen the generation of mockups either on paper or by using specific wireframing tools.

3.3.2 Repeated code. The analysis also highlighted the lack of procedure blocks, that groups a sequence of blocks together, thus avoiding code repetition. The students did not fully understand the functionality of these procedure blocks, and usually grouped all the instructions into a single main block. App Inventor does not provide a step-by-step processing facility for debugging, neither provides a representation of the event queue and a programmer-controlled run-time clock, so users could not see in slow motion how events work. Therefore, students preferred to copy and paste code pieces throughout the program for a better tracking of program execution.

3.3.3 Augmented reality. AR is a technology that, although well known to the students thanks to games, was mostly unknown to them from the point of view of the development of educational resources. Despite that fact, AR was actually applied in several projects. In this vein, we can imagine that other technologies provided by VEDILS that could not be tested in the workshop, such as VR and human-machine interaction with devices, can be within reach of future teachers as developers.

3.3.4 Used elements. Almost all apps contained conditional statements, but conditional loops were rarely used. The list items, which facilitate programming when you want to make batteries of questions, were not used much either, as it has been realized in many applications. We envision that the experience that students will gain from learning based on all the material available on the Web and the development of new applications will lead them to introduce new programming concepts.

4 CONCLUSIONS

In this paper, we have presented the results of a workshop with future teachers as students of a Master's degree. The students, belonging to different educational branches, have been able to test how VEDILS, an extension of App Inventor, enables them to develop educational contents for their future students, without having to be concerned about the programming and coding issues as much as about the educational concepts they want to teach. The apps developed by the students showed that some technologies included in VEDILS, such as augmented reality, can be within everyone's reach. As an ongoing work, this technology is being also tested with professionals of other disciplines, such as health and wellbeing.

The possibility of solving real problems that people from all other disciplines have to face when they become teachers has attracted many of these students to continue working on programming applications. Some of them developed mobile device applications as part of their final Master's thesis and obtained very good qualifications.

ACKNOWLEDGMENTS

This research has been partially funded by the Spanish Ministry of Economy and Competitiveness through the EmPhasys project grant (ref. RTC-2016-5095-1).

REFERENCES

[1] Tedre, M., & Denning, P. J. (2016). The long quest for computational thinking. In Proc. of the 16th Koli Calling International Conference on Computing Education Research, Koli, Finland, pp. 120-129.

[2] García-Peñalvo, F. J. (2016). What Computational Thinking Is. Journal of Information Technology Research, 9(3), 5-8.

[3] Papert, S., & Solomon, C. (1971). Twenty things to do with a computer, Report No. AIM-248, MIT, CSAIL

[4] Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. RED. Revista de Educación a Distancia, 46(10).

[5] Leask, M. & Pachler, N. (2014). Learning to Teach Using ICT in the Secondary School. A companion to school experience, 3rd ed., Routledge.

[6] García-Peñalvo, F. J. (2016). A brief introduction to TACCLE 3 – Coding European Project. In F. J. García-Peñalvo & J. A. Mendes (Eds.), Proc. of 16th Int. Symposium on Computers in Education (SIIE), Sep 13-15, Salamanca, Spain, pp. 1-4.

[7] García-Peñalvo, F. J., Rees, A. M., Hughes, J., Jormanainen, I., Toivonen, T., & Vermeersch, J. (2016). A survey of resources for introducing coding into schools. In F. J. García-Peñalvo (Ed.), Proc. of the 4th Int. Conf. on Technological Ecosystems for Enhancing Multiculturality (TEEM), Nov 2-4, Salamanca, Spain, pp. 19-26.

[8] García-Peñalvo, F. J., Llorens Largo, F., Molero Prieto, X., & Vendrell Vidal, E. (2017). Introducción a la sección especial: Educación en Informática sub 18. ReVisión, 10(2), pp. 13-18.

[9] Lye, S. Y. & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? Computers in Human Behavior, 41(12), pp. 51-61.

[10] Blum, L., & Cortina, T. J. (2007). CS4HS: An outreach program for high school CS teachers. SIGCSE Bulletin, 39(1), pp. 19-23.

[11] Bort, H., & Brylow, D. (2013). CS4Impact: measuring computational thinking concepts present in CS4HS participant lesson plans, Proc. of the 44th ACM SIGCSE, Mar 6-9, Denver, Colorado, USA, pp. 427-432.

[12] Liu, J., Lin, C.-H., Hasson, E. P., & Barnett, Z. D. (2011). Introducing computer science to K-12 through a summer computing workshop for teachers, Proc. of the 42nd ACM SIGCSE, Mar 9-12, 2011, Dallas, TX, USA, pp. 389-394.

[13] Bean, N., Weese, J., Feldhausen, R., & Bell, R. S. (2015). Starting from Scratch Developing a Pre-Service Teacher Training Program in Computational Thinking. IEEE FIE Conference, Oct 21-24, Camino Real El Paso, El Paso, TX, USA, pp. 1307-1314.

[14] Moreno-León, J. & Robles, G. (2016) Code to learn with Scratch? A systematic literature review. Proc. of EDUCON. Apr 10-13, Abu Dhabi, UAE, pp. 150-156.

[15] Wolber. M. (2010). A Blocks Language for Mobile Phones: App Inventor for Android, in E. Canessa & M. Sennaro (Eds.) m-Science. Sensing, Computing and Dissemination, ICTP, pp. 99-128