"Code Yourself" and "A Programar": a bilingual MOOC for teaching Computer Science to teenagers

“Code Yourself” and “A Programar”: a bilingual MOOC for teaching Computer Science to teenagers

Inés Friss de Kereki
Engineering School
Universidad ORT Uruguay
Montevideo, Uruguay
kereki_i@ort.edu.uy

Areti Manataki
School of Informatics
University of Edinburgh
Edinburgh, United Kingdom
A.Manataki@ed.ac.uk

***Abstract*—In a technology-fueled world, coding is an essential skill for young people. MOOCs (Massive Open Online Courses), which are free online courses available to a very large number of people, are an effective and increasingly popular option for teaching scientific topics to a worldwide audience. However, despite the large number of MOOCs available on computer science, there is a scarcity of coding-related MOOCs that are designed for children and teenagers. In this paper, we present a programming MOOC that was recently developed by The University of Edinburgh and Universidad ORT Uruguay for teenager high school students with no prior programming experience. The MOOC was collaboratively developed by the two teams, resulting in a shared instructional design but with a bilingual delivery: “Code Yourself” in English and “A Programar” in Spanish. In this paper, we describe the course design for a young audience and we discuss the international co-development of the course materials. Furthermore, we present results from its simultaneous bilingual delivery in spring 2015, where around 85000 students participated. Student surveys show encouraging results: more than 93% found that the course met or exceeded their expectations, and more than 90% stated that they plan to continue programming in the future.**

Keywords— MOOC; Computer Science 0; Scratch; K-12

I. INTRODUCTION

Over the past few years there has been a growing interest in promoting and teaching computer programming to a worldwide audience through MOOCs (Massive Open Online Courses) [1], [2]. A MOOC is a course of study made available over the Internet without charge to a very large number of people [3]. A wide range of MOOCs in Computer Science (CS) are now available [4]. However, there is a scarcity of computer programming MOOCs that are addressed to children and teenagers. We regard this to be an important gap, especially given the worldwide-recognized need to equip the new generation with coding and computational thinking skills [5].

We address this gap through the development of “Code Yourself” [6] and “A Programar” [7], a two-version MOOC that introduces teenagers to programming and computer science. These free online courses have been jointly developed by The University of Edinburgh and Universidad ORT Uruguay, and they share a common instructional design, while their delivery is in English and Spanish, respectively. The innovation of this project lies in i) the course being designed specifically for teenagers, ii) its international co-development and iii) its simultaneous bilingual delivery in spring 2015.

In this paper, we discuss these topics, reflect on decisions made and share lessons learned. We begin by providing background information on coding skills, computational thinking and MOOCs (Section II). Next, we describe the design of our course, and we discuss international and cultural aspects involved in developing bilingual educational content across two continents (Section III). Finally, we present results from the first course delivery in March-April 2015, with a focus on student demographics, their engagement with the course materials and their evaluation of the course (Section IV). We conclude by discussing lessons learned (Section V) and directions for future work (Section VI).

II. BACKGROUND

A. Computational Thinking and Coding Skills

Historically, computing education started in universities and colleges. Next, it was introduced in high-schools, followed by middle schools [8]. In recent decades, however, Information and Communications Technology (ICT) curriculum in K-10 has typically focused on digital literacy, approaching computing as a tool rather than a science, and hence topics around programming and algorithm development have been largely neglected [9]. Over the last few years, a growing number of educators, as well as parents, economists and politicians in Europe and worldwide are starting to recognize the importance of computational thinking and coding skills [5].

Computational thinking (CT) is a term coined by Jeannette Wing and it involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science [10]. CT is a problem-solving process that includes, among others: formulating problems in a way that enables us to use a computer and other tools to help solve them; logically organizing and analyzing data; representing data through abstractions such as models and simulations; automating solutions through algorithmic thinking; identifying, analyzing, and implementing possible solutions, with the goal of achieving the most efficient and effective combination of steps and resources; generalizing and transferring this problem

The development of "A Programar" was supported by "Santander Universidades”.

solving process to a wide variety of problems [11]. As such, computational thinking means more than being able to program a computer – it requires thinking at multiple levels of abstraction [12]. Computational thinking is a fundamental skill for everyone, not just for computer scientists [10],[13]. It is a transferrable skill, and it is pervasive, influencing research in nearly all disciplines [14]. It has been argued that "computational thinking will influence everyone in every field of endeavor" [12]. This vision poses a new educational challenge for our society, especially for our children.

Coding has also been recognized as a key competence, which will have to be acquired by all young students. Coding skills help to understand today's digitalized society and foster 21st century skills like problem solving, creativity and logical thinking [5]. Programming is not only a fundamental skill of CS and a key tool for supporting the cognitive tasks involved in CT but a demonstration of computational competencies as well [15]. "Contemporaneously, jobs requiring solving unstructured problems, communication, and non-routine manual work have grown as a proportion of the labor market. Under these circumstances, young people's preparation for the workforce must adapt" [16]. It has also been argued that "to function in society, every citizen in the 21st. century must understand at least the principles of computer science" [17].

There is an increasing interest in including computer science topics in the curricula at K–12 level. The Computer Science Teachers Association in the US and Canada, for instance, has developed a set of learning standards for computer science, which involve the following: to introduce the fundamental concepts of computer science to all students, beginning at the elementary school level; to present computer science at the secondary school level in a way that can fulfill a computer science, math, or science graduation credit; to encourage schools to offer additional secondary-level computer science courses that will allow interested students to study facets of computer science in more depth and prepare them for entry into the work force or college [17].

There are several countries around the world where programming or CS has been or will be introduced into secondary but also early childhood education [5],[18]. England is an exemplary case, as it was one of the first European countries to mandate computer programming in its primary and secondary education in state maintained schools from September 2014 onwards. The newly introduced curriculum for computing in England aims to ensure that all pupils: a) can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation; b) can analyze problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems; c) can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems; and d) are responsible, competent, confident and creative users of ICT [19].

B. Programming Languages and Initiatives for Youngsters

Initiatives and online resources that help younger students learn how to code are currently on the rise. Code.org [20], for instance, offers inspirational videos and short lessons that prompt youngsters to create simple programs by plugging generic programming blocks together. The Codecademy [21] online interactive platform, on the other hand, offers coding classes on widely-used programming languages, such as Java and Python.

Educational programming languages that are particularly designed for younger audiences are also increasingly popular. Representative examples include Scratch [22], Alice [23] and AppInventor [24]. These programming languages are typically fully visual, allowing for "drag-and-drop programming", where users build programs by dragging blocks from a block palette and snapping them together with other blocks like a jigsaw puzzle. The resulting structures of multiple blocks correspond to entire scripts or programs. These languages are particularly beneficial for younger students because they require less typing, they minimize syntax errors and they are more engaging and visual compared to industry-standard, textual based languages [25].

Scratch is one of the most successful "drag and drop" programming languages, and it is currently being used in a growing number of schools and code clubs worldwide [5]. Scratch was developed by the Lifelong Kindergarten Group at the MIT, and it is a free educational programming language designed especially for ages 8 to 16 [22]. It was created in order to make it easy and fun for everyone, of all ages, backgrounds, and interests, to program their own interactive stories, games, animations and simulations, and share their creations with one another [26]. Scratch includes an online community with over 11 million registered users and more than 14 million shared projects.

However, the use of Scratch alone may be not sufficient for children to gain a deep understanding of computer science concepts and to develop well-grounded computational thinking and software engineering skills. MOOCs may be particularly useful in this context.

C. MOOCs

"A MOOC is an online course with the option of free and open registration, a publicly shared curriculum, and open-ended outcomes. MOOCs integrate social networking, accessible online resources, and are facilitated by leading practitioners in the field of study" [27]. A MOOC enables students to access high quality academic content. It uses a variety of online resources, such as videos and message boards [28].

There is a growing number of universities worldwide offering MOOCs. This includes renowned American universities, like Stanford, the MIT and Harvard, as well as European organizations, such as the University of Edinburgh [29],[30]. MOOCs are typically hosted in appropriately designed platforms, such as edX, Coursera and FutureLearn. edX is an online learning destination and MOOC provider, founded by Harvard University and the MIT in 2012 [31]. Coursera is a social entrepreneurship company partnering with Stanford, Princeton and other universities, offering currently more than 1,500 MOOCs [32]. FutureLearn is the first UK-led MOOC platform, and it was founded by The Open University

in 2012 [33]. MOOCs are offered in a wide range of languages, and currently there is an increasing number of language-specific MOOC platforms. For instance, Universia and Telefónica Learning Series created a Spanish platform, called Miríada X, where several Latin American Universities provide MOOCs [34].

The MOOC landscape is still evolving. Setting up MOOCs requires, among other elements, a good understanding and appreciation of learning theories and e-pedagogical strategies. Various learning theories (as behaviourism, cognitivism, constructivism, connectivism, etc.) "all have something to offer in the consideration of e-strategies for MOOCs" [35], and Coelho refers especially to constructivism and connectivism [36]. The Community of Inquiry framework for online learning is also of relevance to MOOCs [37]. In this, the educational experience is shaped by three elements: social, cognitive and teaching presence. Social presence is the ability of learners to project themselves socially and emotionally, and it can be cultivated through collaborative activities, among others. Cognitive presence is described as the extent to which learners are able to construct and confirm meaning through sustained discourse, and it can be supported through appropriate scaffolding for online discussions. Finally, teaching presence involves design and organization (e.g. being clear about the course structure, deadlines, assessment, etc.), facilitating discourse and direct instruction. Apart from pedagogical theory, there is also great interest in best practices for developing successful courses. Some lessons learned from existing MOOCs include keeping courses short (ideally up to four weeks, as longer courses lead to more dropouts), keeping the videos short (less than six minutes), and having course instructors project an informal air [38]. It has also been found that students generally engage more with videos where instructors speak faster [39]. Furthermore, including different types of videos, such as talking head, animated slide presentations, tablet capture or combinations, is considered to be a good practice [40]. Including frequent, short quizzes interspersed with content can also enhance student engagement [41]. As far as assessment is concerned, it has been recommended that quizzes should give feedback, deadlines should be clearly specified and, in the case of peer-assessment, guidelines and rubrics with clearly defined tasks should be made available to the reviewer [42].

D. MOOCs in Computer Science

There is a plethora of introductory Computer Science MOOCs. The majority of these teach professional programming languages and are targeted to adults, while only a small number of courses are designed for younger audiences. A MOOC within the first category is "CS50x: Introduction to Computer Science", which is offered by Harvard University [43]. This is a 12-week, entry level course that teaches students how to think algorithmically and solve problems efficiently. Topics covered include, among others, abstraction, algorithms, data structures, encapsulation, resource management and security. Different programming languages are presented in the course, such as C, PHP, JavaScript, CSS, HTML and Scratch. In our opinion, CS50x is a stimulating course, reflecting the breadth and richness of computer science. However, it might be challenging for youngsters to follow, as the level of difficulty increases quickly throughout the course.

"Intro to Computer Science" is another MOOC that uses a professional programming language, in particular Python [44]. This course has a duration of 3 months and it is centered around learning how to build a search engine and a social network. The course provides "a good level of scaffolding, so the student is not challenged to build a program from scratch" [45]. It does not include sufficient coverage of software engineering practices, which is central to successful software development [45].

An example of a MOOC designed for youngsters is "MyCS: Computer Science for Beginners", which is offered by Harvey Mudd College [46]. This course is an early introduction to CS, exploring a combination of how computers work and how we can use them to solve interesting problems. It lasts five weeks, throughout which lessons alternate between general CS topics and Scratch programming activities. The course is intended especially for middle school students and their teachers, but it is good for students of all ages [46]. However, it includes relatively few videos and a lot of text guides, and, as mentioned in the syllabus, "there aren't as many videos as you might expect in more advanced CS courses, nor are the videos as high in production quality as other series you might see". This might make the course content less engaging for students, particularly young ones.

"Programming in Scratch" is another MOOC offered by Harvey Mudd College that is oriented to kids. As explained in the course syllabus, "it has ten main lessons, each of which should take about three hours to complete. Each lesson introduces some new concepts in Scratch by way of a mix of videos, text instructions, and practice questions. After the new concepts are introduced, you'll have a homework assignment and a quiz to complete." [47]. This 6-week course provides a thorough introduction to programming with Scratch, but it doesn't cover more general CS topics, such as computational thinking or software engineering, thus raising questions about the transferability of the skills developed. Furthermore, the course is limited to English. For instance, in the final project, no marks are given to answers that are not in English. There is also no peer evaluation.

In Spanish, we could refer to the 5-week MOOC, "Computational Thinking in School", which presents the main ideas of computational thinking [48]. The course consists of two parts: i) a conceptual introduction to the ideas behind computational thinking and their application in everyday environment, and ii) a practical introduction to the implementation of computational thinking using Scratch. Even though this course enables a deep reflection on computational thinking concepts, its video content may be less engaging for youngsters, as it consists mostly of images with recorded voices, rather than animated videos. The code samples in Scratch are presented, in some cases, first completed and then, that code is split in their components and rebuilt from these components. In our opinion, this approach does not follow the typical problem-solving process that students need to develop.

SM4T (Scratch MOOC for Teens) is another Spanish-speaking MOOC that was developed in 2013 by Universidad ORT Uruguay and Plan Ceibal, which is Uruguay's one laptop per child -and per teacher- program for public schools and high schools in the country [49]. This 5-week MOOC was specially designed for teenager high school students in Uruguay. Its aim was to promote the development of procedural thinking and problem-solving skills through learning the basics of computer programming using Scratch. The course included short videos, work guides ("recipes" to obtain certain results), self-assessment exercises and Scratch code samples. The course dynamics included showcasing example video games developed using Scratch and then introducing the components needed to build these programs. Different programming concepts were included transversely, as required to resolve or include game functionalities [49]. The course did not include software engineering practices.

To sum up, there are different options for teaching and learning introductory computer science in a MOOC format. According to Adamopoulos, the more satisfied a student is with the professor, the teaching material and the assignments, the more probable he/she is to successfully complete the course [50]. The author also found that projects make the course more engaging, and that peer assessment has a more positive effect on course completion compared to automated feedback. Given these points, we believe that important characteristics for computer science MOOCs, and particularly, teenager-oriented courses, are: interesting and fun topics and challenges, tasks difficulty in accordance with the course level, gradually increasing complexity of tasks, different types of evaluation (including peer evaluation), and attractive and well-designed videos. In our opinion, none of the presented courses included all these characteristics.

III. COURSE DESIGN

A. Objective and description

In order to address this gap, we have developed "Code Yourself" [6] and "A Programar" [7], a two-version MOOC that is delivered in English and Spanish, respectively. Our objective with this MOOC is to teach the fundamentals of computer programming to teenagers worldwide, while promoting the development of computational thinking and the use of basic practices in software engineering. No previous programming experience is required. By the end of the course, students are expected to:

- Understand and apply fundamental principles and concepts of computer science;
- Analyze problems in computational terms;
- Understand and follow basic software engineering practices; and
- Design, create, debug, reuse and re-purpose computer programs in Scratch.

We have decided to use Scratch because it works particularly well with youngsters, as its visual nature allows them to abstract from syntax and, instead, focus on computational thinking. Our aspiration with this MOOC is that it will equip young students with a solid understanding of the foundations of computing, allowing them to move to different programming languages or take a more advanced programming course in the future.

The curriculum looks as follows:

- Unit 1: Your First Computer Program: Introduction to Algorithms; How to Use Scratch; Introduction to Selection; Iteration; Simple games and animations with Sequence, Selection and Iteration
- Unit 2: Code Gone Loopy!: Loops; Event-driven Programming; Abstraction and Decomposition; Software Requirements, Design and Implementation
- Unit 3: Remixing Games: Variables; Complex Conditions; Nested Loops; Software Testing and Documentation
- Unit 4: Reusing Your Code: Procedures; Generalization; Cloning; Modularity and Flexibility
- Unit 5: Think Like a Software Engineer: Software Development Approaches; Coordination; Zombie Game; What Next?

In order to make the curriculum appealing to youngsters, each unit is presented from a practical angle, as a series of fun challenges to be tackled when building games and animations in Scratch. For instance, Unit 3 includes remixing a game, originally built in the previous unit, in which the player navigates a helicopter through a sky full of clouds. The game extension involves counting the number of times the helicopter touches a cloud, for which we need to introduce the concept of variable. Hence, the main narrative of the course is around guiding students to create interesting programs, while the theoretical concepts are introduced as required to include program functionality. Furthermore, open challenges are included in each unit, so as to foster creativity and experimentation when programming.

The MOOC lasts 5 weeks and contains approximately 1 hour of weekly lectures in the form of short videos (3-5 minutes), supported by texts and Scratch code examples. Additionally, optional in-video quizzes are embedded in the lectures, so as to help students verify their understanding of the concepts taught, as well as enhance student engagement [41]. Fig. 1 presents a video snapshot of "Code Yourself". Interviews with experts in computer science and other fields are also included to reflect on the topics covered each week and show links with other facets of real life. In Fig. 2 there is a snapshot of the materials in "Code Yourself" (the materials in "A Programar" are similar). The options shown on the right side of each video link are: forum access, code samples, video downloads and transcripts.

Assessment is based on quizzes and peer-reviewed projects, which enable students to acquire practical programming experience. This approach allows for richer evaluation, as recommended by Cooper and Sahami [51]. There are 5 multiple choice quizzes, worth 10 points each. Students can attempt each quiz at most 5 times and the final grade awarded is the highest grade achieved. All quizzes

include multiple choice or true/false questions and give feedback for incorrect answers. There are 2 peer-reviewed projects, worth 20 and 30 points respectively. The first project involves creating an animation that introduces two characters and a place of interest. It includes the application of basic concepts like algorithm, sequence and selection. The second project is a game called "Homesick Cody", where the student helps Cody (an alien character used in the course) to go back to his planet. This game includes almost all concepts presented in the course. Apart from submitting their own projects, students are expected to evaluate the work of at least 3 of their classmates. Clear directions and rubrics are provided for this task, as recommended by Yousef et al [42]. Peer assessment brings pedagogical, metacognitive and affective benefits [52], supporting the social presence of students and allowing them to compare different approaches and employ critical judgment. Students are also prompted to include their own comments for improving the game reviewed. Students successfully complete the course when they obtain at least 50% of the available points. Upon successful completion, they receive a "statement of accomplishment" (certificate) issued by Universidad ORT Uruguay and The University of Edinburgh, signed by both instructors.

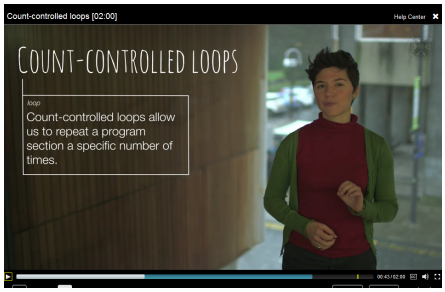

Fig. 1. Snapshot of a "Code Yourself" video

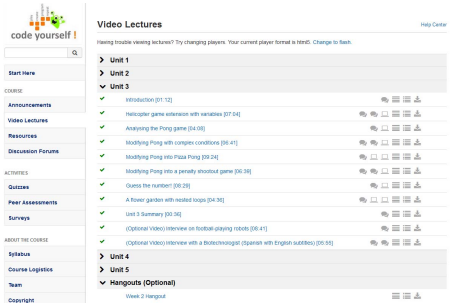

Fig. 2. Snapshot of "Code Yourself" materials

As far as communication is concerned, two main channels are used: discussion forums and emails. Discussion forums allow students to interact with each other and learn together, thus supporting their social and cognitive presence, as conceptualized in the Community of Inquiry framework. We encourage forum participation by inviting students to introduce themselves in the social forum, as well as by including in the videos some questions to be discussed in the forums. Some guidelines around the use of the forums is also included in the course website. To promote student-faculty contact, weekly mails and announcements are also included.

B. Co-development of the course and considerations

The educational content was designed and developed together by Universidad ORT Uruguay and The University of Edinburgh. ORT's team included a General Coordinator, three Computer Science teachers, two platform Specialists, four Audio Visual Specialists, four Multimedia Specialists, and one expert in Educational Evaluation. The University of Edinburgh team was similar. We also had the support of Coursera Specialists, as the MOOC was developed for the Coursera platform [32].

This international collaboration was enabled through regular virtual meetings, during which we defined the instructional design and discussed ideas. After developing some initial ideas remotely, we organized a face-to-face 3-day meeting in Edinburgh, which allowed us to decide on the final shape of the course. This visit consolidated the team and made the communication more fluent, allowing for an even smoother collaboration thereafter.

During the design and creation process, we versioned the documents carefully and we used a common document sharing platform, which allowed us to collaborate and co-develop the materials. Our original plan was to first develop the material fully in English and then translate it to Spanish. During the first weeks of our collaboration, however, we noticed that it would be better to develop the material together in a mix of English and Spanish, to allow each teacher to write faster in their own mother tongue. The teachers speak both languages. The final versions in each language were written at the very end, when we agreed on all the concepts, examples, quizzes, and all the details to be used.

We took into account the target audience when designing the course content. We tried to create an international and open course with no differences (gender, geographical considerations, social context, prior knowledge, etc.). Materials were designed so as to be attractive and relevant to young people. We considered using examples interesting for boys and girls, teens and also adults, and we included different types of games and animations. Some examples are: the classic Pong game with two variations, "Pizza Pong" and "Shooting penalties"; drawing a garden of flowers; and a more advanced game, called "Chasing Zombies", which has multiple characters and backgrounds.

We developed together the complete set of scripts and discussed in detail each proposal and example. When designing learning experiences and assessments, we considered the revised version of Bloom's taxonomy which considers two dimensions: cognitive process and knowledge [53]. The cognitive process dimension moves from remembering to understanding, applying, analyzing, evaluating and creating. The knowledge dimension includes factual, conceptual, procedural and metacognitive knowledge. In our case, the assignments are reflective and encourage deep analysis, which is recommended by Bali [54]. In particular, peer review activities involve applying, evaluating and some level of metacognitive learning. Quizzes involve factual and conceptual knowledge. They focus not only on remembering and understanding, but they also include questions that promote deep reflection.

Based on our previous experience presented in [49], we wrote the entire collection of scripts, which we divided in "videos" rather than "chapters". This division was very useful for predicting the duration of each video. We used visual resources, such as drawings, animations, images and fonts, which were developed by the two teams and shared. The final versions of all text and visual resources were checked by our communication specialists, who gave us valuable suggestions for refining and adjusting them to account for a young and worldwide audience, from different countries, cultures and backgrounds.

After finalizing the texts, we recorded the videos. In this process, we used a teleprompter to allow the teachers to be more comfortable during the recording. All the teachers (including teaching assistants) transmitted energy and enthusiasm, making direct eye contact with the camera. We also used discrete make-up and non-distracting clothes, opting for long sleeve shirts for cultural reasons. We selected interesting but non-distracting backgrounds to make the experience of the course better: an interior space with exterior view that does not distract students. In particular, we chose a background through a window where the perceived elements form a whole and none of them stands out or captures our attention, so that the central point of focus is the teacher. In some units we decided to record with a rainy background and others were recorded on a sunny day, so as to distinguish between them. This was also achieved by wearing different clothes for different units. In the Spanish version we filmed mainly in an office with sunny and rainy Uruguayan coast side background. In the English course we chose mostly offices and beautiful spaces on the university campus. Following the recommendation by Alario et al [40], we included videos of different types: talking head, animated slides, stop motion animations, drawings, and screen captures when coding in Scratch.

Furthermore, we decided to include an animated alien character that asks interesting questions during the videos. This character was also included in the exercises. Its name, "Cody", was chosen considering both languages. In Fig. 3 there is a snapshot of the instructor in "A Programar"' and Cody.

Fig. 3. Snapshot of an "A Programar" video: instructor and Cody

The videos were similar in the two languages. The only difference in the final videos was around the use of music. In the Spanish version we included a short characteristic tune at the beginning and the end of each video, while in the English version this was not included. It is also worth noting that closed captions were included for all videos in both languages.

All the interviews in English and Spanish were included in both courses, thus reinforcing the idea that "Code Yourself" and "A Programar" are essentially different versions of the same course. Interviewees were selected carefully, so as to cover a wide range of ethnic backgrounds, ages, genders and expertise. The objective was to relate the principles of computer science to different areas, for instance biotechnology, robotics or even jewelry making. The video settings were also carefully selected, for example in the referred interviews they were: a biochemistry laboratory, a robotics lab and an actual jewelry workshop.

"Code Yourself" and "A Programar" materials were pretested with different groups of students in both the United Kingdom and Uruguay. We showed students course videos and Scratch examples, and asked them for comments and recommendations. Their proposals were included in our final versions.

IV. COURSE DELIVERY – 1ST. SESSION

This bilingual MOOC was first delivered during 9th March- 12th April 2015. Both "Code Yourself" and "A Programar" were launched simultaneously. The promotion of the course was made via Facebook, Google AdWords, and academic networks. Table I presents the main statistics related to enrollment, engagement and course completion. More than 4/5 of the students indicated that they have no CS background.

TABLE I. 1ST SESSION GENERAL DATA

1st session		
Topic	***"Code Yourself"***	***"A Programar"***
Enrolled	59,531	25,255
Previous CS knowledge	82% no	89% no
Completed initial survey	6229	6429
Earned certificate	1595	1592
Visited the course at least once	37048	16780
Watched at least one video	26190	13840
% certificate earned /enrolled	2.68%	6.30%
% certificate earned /visited the course at least once	4.31%	9.49%
% certificate earned /watched at least one video	6.09%	11.50%

As shown in Table I, completion rates for "Code Yourself" and "A Programar" are 2.68% and 6.3%, respectively. These are comparable to the average completion rate for Coursera MOOCs, which is 5% [55]. It is also worth noting that the completion rate for two Spanish-speaking MOOCs in CS was 3.99% [56], and, hence, "A Programar" achieved better results.

In Table II we include information related to gender, age and native language. As far as gender is concerned, in CS and Science, Technology, Engineering and Mathematics (STEM) courses, men outnumber women by five to one, on average [29]. In our course, the proportion was almost one to one (in

the English version) and two to one (in the Spanish version). Hence, our course obtained a greater gender inclusion. In "A Programar", 15% of the students were younger than 18 years old. It is a higher percentage compared to "Code Yourself" (9%), but both are higher than the statistics reported by Nesterko et al [57], who mentions that 7.8% of all HarvardX students are from 6 to 20 years old. It is also interesting that in the Spanish version almost all of the students who completed the initial survey are native Spanish speakers. In the English version half of the students are English native.

TABLE II. STUDENT DEMOGRAPHICS (BASED ON INITIAL SURVEY)

1st session		
Topic	***"Code Yourself"***	***"A Programar"***
Gender	Male 54% Female 44% No answer 2%	Male 65% Female 34% No answer 1%
Students<18 years old	9%	15%
Is the language of the course your native language?	51% yes	97% yes

Many participants who enroll in MOOCs never have the intention to complete them at all [55],[58]. Ho et al. mention that 57% of HarvardX and MITx students that responded to a survey stated their intent to earn a certificate, and 24% of these respondents earned certificates [29]. We observed a similar value (25.64%) in "A Programar" but in "Code Yourself" this value was lower (16%). This difference may be related to the students' native language, considering that their CS background was similar (more than 80% of the students have no previous CS knowledge in both courses). However, further research would be needed to verify this claim.

In Table III we provide data related to the geographical distribution of our students. We have 197 different countries in "Code Yourself" and 117 in "A Programar". The different distributions can be explained by the language of each course. For instance, it comes to no surprise that Mexico, Spain and Colombia are amongst the top 5 countries for "A Programar".

We have collected feedback from the students through an optional survey at the end of the course. The main data is presented in Table IV and Figure 4. The vast majority of the students that completed the survey found that the course met or exceeded their expectations (93% "CodeYourself", 95% "A Programar", see Fig 4.), and also would recommend the course to a friend (96-98%). Their future plans to continue programming are also very encouraging: 90% or more indicated this option, as shown in Table IV. In particular, 66% of both courses' participants plan to continue programming not only in Scratch but also in another programming language. Considering "Code Yourself" and "A Programar" respectively, the pacing of the course was rated as "right" by 81% and 79% of the students, length as "just right" by 72% and 77%, and difficulty as "just right" by 75% and 85%. The most valuable elements were the video lectures (referred by 93% and 96%), quizzes (54% and 65%) and peer assessments (52% and 60%).

TABLE III. GEOGRAPHICAL DISTRIBUTION OF STUDENTS

1st session		
Topic	***"Code Yourself"***	***"A Programar"***
Countries	197	117
Continents	North and Central America: 37%, Asia: 28%, Europe: 25%, Africa: 5%, South America: 4%, Rest of the world: 1%	South America: 40%, North and Central America: 35%, Europe: 23%, Asia:1,6%, Rest of the world: 0,4%
Top 5 Countries	United States: 31%, India: 13%, United Kingdom: 4%, China: 4%, Canada: 3%	Mexico: 20%, Spain: 20%, Colombia: 12%, United States: 7%, Uruguay: 6%

TABLE IV. EVALUATION AND FEEDBACK FROM STUDENTS

1st session		
Topic	***"Code Yourself"***	***"A Programar"***
Completed final survey	896	1587
Would you recommend this course to a friend?	96% yes/ probably	98% yes/ probably
Do you plan to continue programming in the future?:	95% yes	90% yes
a) "Yes, I plan to continue programming in Scratch as well learn another programming language"	66%	66%
b) "Yes, I plan to learn another programming language, but I do not plan to continue programming in Scratch"	20%	11%
c) "Yes, I plan to continue programming only in Scratch"	9%	13%
How would you rate your overal experience with this course?	89% "very good"/ "excellent"	87% "very good"/ "excellent"

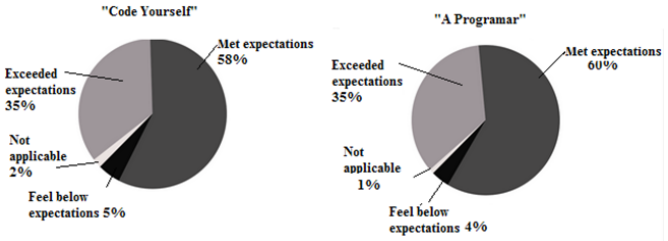

Fig. 4. Did you get what you wanted from the course?

Some quotes from "A Programar" participants follow: "I loved the way the teacher explained things, the quality of the videos, the exercises, how Software Engineering concepts were explained through something as simple as a game in Scratch."; "It was the best course I have taken, honestly, resources and evaluations were adapted to the program", and "Many congratulations on your work. Good course with gradual increase in difficulty and the level of programming. Good evidence of knowledge and good work in the forums.". Feedback from "Code Yourself" participants includes the following: "Outstanding quality of teaching"; "I was amazed

of how well the course and the material was developed and taught. Brilliant!"; "It was an exciting course and I certainly learnt a lot from it"; and "Thanks for the wonderfully designed course and introducing me to programming."

V. LESSONS LEARNED

In this section we discuss lessons learned from the first delivery of the course, including challenges and approaches for tackling them. For instance, we noticed that some students reported technical difficulties related to their user account on the Coursera platform. After investigating this further, we realized that some students registered twice or more times for the course: they used their Facebook, Google or other accounts. To try to solve this, we put some comments in the forums.

We also assumed that students would be competent with regards to downloading, saving or uploading files. However, there was a large number of questions about these topics in the forums. Therefore, we developed and included new videos with thorough explanations: how to download/upload a file, how to save a file, etc.

Another frequently asked question was related to verified certificates. In the Coursera platform, there were two options at that time: a free unverified identity certificate and one that is paid-for, with verified identity (through webcam and writing pattern recognition). We explained this point in the forums to avoid confusion. We also noticed that some students did not check in the course specification which activities were compulsory and which were optional. We, thus, had to put more emphasis on these differences and highlight that forum participation was optional.

With regards to forum use, we noticed that many students did not follow a particular thread, but instead they created a new one for the same question or topic. We tried to minimize this problem of redundancy by moving some threads around and including more guidelines. We decided to keep teacher participation in the forums relatively low. Our rationale for this was that we didn't want to monopolize discussions, and instead wanted to shift the focus to the students, promoting student collaboration. Most of the time we did not reply to questions directly, but instead we waited for a response from other students. Only in special cases did teaching staff answer a question, particularly when a wrong answer was give or a misconception was discussed. We believe that this approach worked well, as discussions in the forums were vibrant and we could sense a strong community feeling among students.

As explained in Section III-B, we took into account the target audience when developing the course material. One point that we missed was color blind people. A student suggested that we should avoid examples that refer only to particular colors. An example of this is a game, in which the player is meant to move a cat to touch a green, orange or violet balloon.

VI. CONCLUSIONS AND FUTURE WORK

In this work, we presented a bilingual MOOC that introduces teenagers to programming and computer science. We described the course design and development, and we presented results from its first delivery in 2015. Given the student feedback and results, we evaluate this experience as very good. The vast majority of students stated that the course met or exceeded their expectations and that they plan to continue programming (in Scratch and other programming languages) in the future. This is remarkable, as one of our aspirations with this course was to provide students with a solid grounding in computing that would inspire them to develop their programming skills further. It is worth noting that for both course development teams, the MOOC experience was exciting and challenging. We also found that the international and collaborative element of this project helped create a course of high quality, which would have not been achieved by a single team. In this paper, we have discussed aspects of the co-design and co-development experience, and we have shared lessons learned, which can be useful for other teams that decide to develop MOOCs in a collaborative fashion.

"Code Yourself" and "A Programar" are currently being offered on Coursera in an "auto-cohort" mode, which means that new sessions begin every month. The overall material is almost the same as previously, while some resources have been improved based on the experience of the first session (e.g. new detailed guides on how to manage Scratch files have been included). Furthermore, course mentors are now included in the new format. These are high-performing students from the first session, who have been invited to attend the new, auto-cohort sessions and support the students in the forums.

As future work, we plan to analyze and compare the results of the course in the new mode with the previous one and the long-term impacts. Moreover, we are working on a Portuguese version of the course, called "Programe-se".

REFERENCES

[1] N. Spyropoulou, G. Demopoulou, C. Pierrakeas, I. Koutsonikos, and A. Kameas, "Developing a Computer Programming MOOC". Procedia Computer Science, 65, 182-191, 2015.

[2] D. Malan, "Implementing a massive open online course (MOOC)", Journal of Computing Sciences in Colleges table of contents archive, Volume 28 Issue 6, p. 136-137, June 2013.

[3] Oxford Dictionary, http://www.oxforddictionaries.com/us/definition/american_english/MOOC. Accessed December 15th, 2015.

[4] M. Guzdial, and J. Adams, "MOOCs need more work; so do CS graduates", Communications of the ACM CACM, Volume 57 Issue 1, Pp 18-19, January 2014.

[5] A. Balanskat, and K. Engelhardt. (contributors), "Computing our future. Computer programming and coding. Priorities, school curricula and initiatives across Europa". European Schoolnet, Belgium. 2014.

[6] MOOC: "Code Yourself". The University of Edinburgh and Universidad ORT Uruguay. https://www.coursera.org/learn/intro-programming. Accessed December 15th, 2015.

[7] MOOC: "A Programar". Universidad ORT Uruguay and The University of Edinburgh. https://www.coursera.org/learn/a-programar. Accessed December 15th, 2015.

[8] M. Armoni, "Early Education – what does computing has to do with it and in what ways?", WiPCSE'15 (Proc of the Workshop in Primary and Secondary Computing Education), ACM, USA, 2015.

[9] K. Falkner, R. Vivian, and N. Falkner, "Teaching Computational Thinking in K-6: The CSER Digital Technologies MOOC", ACE 2015, Sydney, Australia, 2015.

[10] J. Wing, Computational thinking. Communications of the ACM, Vol. 49, No. 3, March 2006.

[11] CSTA: Operational Definition of Computational Thinking for K-12 Education, https://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf Accessed March 14 th, 2016.

[12] J. Wing, Computational thinking and thinking about computing. Phil. Trans. R. Soc. A 366, 3717–372, 2008.

[13] Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., and J. T. Korb, . 2014. Computational thinking in elementary and secondary teacher education. ACM Trans. Comput. Educ.14, 1, Article 5, March 2014.

[14] A. Bundy, "Computational thinking is pervasive". J. Sci. Pract. Comput. 1, 67–69, 2007.

[15] S. Grover, and R. Pea, "Computational Thinking in K–12. A Review of the State of the Field", Educational Researcher, vol. 42 no. 1 38-43, Jan/Feb 2013.

[16] I. Lee, F. Martin, and K. Apone, "Integrating Computational Thinking Across the K-8 Curriculum", ACM Inroads Inroads, V 5 Issue 4, Pp. 64-71, ACM New York, NY, USA, Dec. 2014

[17] ACM Computer Science Teachers Association. CSTA K–12 Computer Science Standards (2011) http://csta.acm.org/Curriculum/sub/K12Standards.html. Accessed December 15th, 2015.

[18] L. Mannila, V. Dagiene, B. Demo, N. Grgurina, C. Mirolo, L. Rolandsson, and A. Settle,"Computational Thinking in K-9 Education", ITiCSE-WGR'14, Uppsala, Sweden, 2014.

[19] Department for Education, "National curriculum in England: computing programmes of study (2013)". https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study. Accessed Dec 15th, 2015.

[20] Code.org, https://code.org/ Accessed December 15th, 2015.

[21] Codecademy,http://www.codecademy.com/ Accessed Dec 15th, 2015.

[22] Scratch, wiki.scratch.mit.edu/wiki/Scratch Accessed Dec 15th, 2015.

[23] Alice, http://www.alice.org. Accessed January 4th, 2016.

[24] MIT AppInventor, appinventor.mit.edu. Accessed Dec 15th, 2015.

[25] S. Esper, S. Foster, W. Griswold, C. Herrera, and W. Snyder, "CodeSpells: Bridging Educational Language Features with Industry-standard Languages", Koli Calling 2014, Koli, Finland, 2014.

[26] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E.Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai. "Scratch: Programming for All. Communications of the ACM", Vol. 52 No. 11, Pages 60-67, 2009.

[27] A. Mc Auley, B. Stewart, G. Siemens, and D. Cormier, "The MOOC Model for Digital Practice". http://davecormier.com/edblog/wp-content/uploads/MOOC_Final.pdf Accessed December 15th, 2015.

[28] Universities UK: "Massive open online courses. Higher education's digital moment?" www.universitiesuk.ac.uk/highereducation/Documents/2013/MassiveOpenOnlineCourses.pdf. Accessed December 15th, 2015.

[29] A. Ho, I. Chuang, J. Reich, C. Austun Coleman, J. Whitehill, C. Northcutt, J. Williams, J. Hansen, G. Lopez, and R. Petersen, "HarvardX and MITx: Two Years of Open Online Courses Fall 2012-Summer 2014". Available at SSRN: http://ssrn.com/abstract=2586847 or http://dx.doi.org/10.2139/ssrn.2586847 Accessed March 21th, 2016

[30] University of Edinburgh, http://www.ed.ac.uk/studying/online-learning/moocs/moocs. Accessed December 15th, 2015.

[31] edX, https://www.edx.org/. Accessed December 15th, 2015.

[32] Coursera, https://www.coursera.org/. Accessed December 15th, 2015.

[33] FutureLearn, www.futurelearn.com/courses. Accessed Dec 5th, 2015.

[34] Miríada, https://miriadax.net/home. Accessed March 21th, 2016.

[35] E. O'Donnell, M. Sharp, S. Lawless, and L. O'Donnell, "Learning Theories: ePedagogical strategies for MOOCs in Higher Education", in: "Macro-Level Learning through MOOCs: Strategies and Predictions for the Future", pp. 92-118, Hershey, PA, USA, 2015

[36] D. Coelho, "Learning Theories Supporting Massive Open Online Courses", in: "Furthering Higher Education Possibilities through Massive Open Online Courses" (A. Mesquita, and P. Peres, editors), pp. 125-149,Hershey, PA, USA, 2015

[37] D. R. Garrison, T. Anderson, and W. Archer, "Critical inquiry in a text-based environment: Computer conferencing in higher education". The Internet and Higher Education, 2(2-3), 87-105, 2000.

[38] K. Button, "10 Lessons learned from Moocs", Education Dive, http://www.educationdive.com/news/10-lessons-learned-from-moocs/306113/. Accesed January 4th, 2016.

[39] P. J. Guo, J. Kim, and R. Rubin, "How video production affects student engagement: an empirical study of MOOC videos", L@S '14 Proceedings of the first ACM conference on Learning @ scale conference, USA, 2014.

[40] C. Alario Hoyos, M. Pérez-Sanagustín, C. Delgado, and P. J. Muñoz-Merino, "Recommendations for the design and deployment of MOOCs: insights about the MOOC digital education of the future deployed in MiríadaX", TEEM '14 Proc. of the Second Int. Conf. on Technological Ecosystems for Enhancing Multiculturality, Pp 403-408, ACM New York, NY, USA, 2014.

[41] W. Siever, "Leveraging MOOCS", JCSC 30, 2, Dec 2014.

[42] A. M. F. Yousef, M. A. Chatti, U. Schroeder, and M. Wosnitza, "What Drives a Successful MOOC? An Empirical Examination of Criteria to Assure Design Quality of MOOCs", Conf. on Advanced Learning Technologies (ICALT), Greece, 2014.

[43] MOOC: "Introduction to Computer Science", Harvard University https://www.edx.org/course/introduction-computer-science-harvardx-cs50x. Accessed January 4th, 2016.

[44] MOOC: "Intro to Computer Science". https://www.udacity.com/course/intro-to-computer-science--cs101. Accesed Jan 4th, 2016

[45] M. Ben-Ari, "MOOCs on introductory programming: a travelogue". ACM Inroads, 4(2), 58-61, 2013.

[46] MOOC: "My CS: Computer Science for Beginners". https://www.edx.org/course/mycs-computer-science-beginners-harveymuddx-cs001x# Accessed December 15th, 2015.

[47] MOOC: "Programming in Scratch". Harvey-Mudd College. https://www.edx.org/course/programming-scratch-harveymuddx-cs002x-0. Accessed December 15th, 2015.

[48] MOOC: "Computational Thinking in the School (2nd. edition)" (In Spanish) https://miriadax.net/web/pensamiento-computacional-en-la-escuela-2ed. Accessed December 15th, 2015.

[49] I. Kereki, and J. V. Paulós, "SM4T: Scratch MOOC for Teens - A pioneer pilot experience in Uruguay", FIE 2014, Spain, 2014.

[50] P. Adamopoulus, "What makes a great MOOC? An interdisciplinary analysis of student retention in online courses". 34th International Conference on Information Systems, Milan, 2013.

[51] S. Cooper, and M. Sahami, "Reflections on Standford's MOOCs", Communications of ACM, V 56, N 2, February 2013.

[52] P. M. Sadler, and E. Good. "The impact of self-and peer-grading on student learning". Educational assessment, 11(1):1-31, 2006.

[53] D. Krathwohl, "A Revision of Bloom's Taxonomy: an Overview", Theory into Practice, V. 41, N 4, USA, 2002.

[54] M. Bali, "MOOC Pedagogy: Gleaning Good Practice from existing MOOCs", MERLOT J. of Online Learning and Teaching, V. 10, N.1, March 2014.

[55] D. Koller, A. Ng, C. Do and Z. Chen, "Retention and intention in massive open online courses: In depth". Educause Rev, 48(3), 62-63, 2013.

[56] "First year of Coursera MOOCs at UC Chile". https://mperezsanagustin.files.wordpress.com/2016/01/informeresumen_moocs_online-enero2016.pdf. Accessed April 4th, 2016.

[57] S. Nesterko, D. Seaton, K. Kashin, Q. Han, J. Reich, J. Waldo, I. Chuang I., and A. Ho, Age Composition (HarvardX Insights), 2014, http://harvardx.harvard.edu/harvardx-insights/age-composition. Accessed March 22th, 2016.

[58] S. Zheng, M. Rosson, P. Shih, and J. Carroll, "Understanding student motivation, behaviors, and perceptions in MOOCs", CSCW 2015, pp. 1882-1895. Canada, 2015.