Evolution of teaching and evaluation methodologies: The experience in the computer programming course at the Universidad Nacional de Colombia

Evolución de las metodologías de enseñanza y evaluación: la experiencia del curso de programación de computadores en la Universidad Nacional de Colombia

J. Gómez[1], E. León[2], C. Cubides[3], A. Rodríguez[4], J. Mahecha[5] and J. C. Rubiano[6]

ABSTRACT

In this paper, we present the evolution of a computer-programming course at the University Nacional de Colombia (UNAL). The teaching methodology has evolved from a linear and non-standardized methodology to a flexible, non-linear and student-centered methodology. Our methodology uses an e-learning platform that supports the learning process by offering students and professors custom navigation between the content and material in an interactive way (book chapters, exercises, videos). Moreover, the platform is open access, and approximately 900 students from the university take this course each term. However, our evaluation methodology has evolved from static evaluations based on paper tests to an online process based on computer adaptive testing (CAT) that chooses the questions to ask a student and assigns the student a grade according to the student's ability.

Keywords: computer adaptive tests (CAT), e-learning platform, teaching methodology, MOOC, computer programming course, experience, evolution.

RESUMEN

En este artículo presentamos la evolución del curso de programación de computadores en la Universidad Nacional de Colombia (UNAL). La metodología de enseñanza evolucionó de una lineal no estandarizada a una metodología flexible, no lineal y centrada en el estudiante. En un lado, nuestra metodología emplea una plataforma de aprendizaje en línea que apoya los procesos de aprendizaje ofreciendo a los estudiantes y profesores navegación autónoma de contenidos y materiales en una forma interactiva (capítulos de libro, ejercicios y videos). Además, la plataforma tiene acceso libre y cerca de 900 estudiantes toman este curso por semestre. Por otro lado, nuestra metodología de evaluación ha evolucionado de realizar exámenes conjuntos basados en papel a evaluaciones en línea adaptativas usando el concepto de CAT ((Pruebas adaptativas basadas en computador), que selecciona las preguntas a realizarle al estudiante y que asigna la calificación del estudiante, dada su habilidad.

Palabras clave: evaluación adaptativa, plataforma de aprendizaje en línea, metodología de enseñanza, MOOC, curso de programación de computadores, experiencia, evolución.

Received: February 24th 2014
Accepted: March 20th 2014

[1] Jonatan Gómez. Computer Systems Engineer, Universidad Nacional de Colombia. Ph.D. in Mathematics, Computer Science concentration, The University of Memphis. Affiliation: Associated Professor, Universidad Nacional de Colombia.
E-mail: jgomezpe@unal.edu.co

[2] Elizabeth León. Computer Systems Engineer, Universidad Nacional de Colombia. Ph.D. in Computer Science and Computer Engineering, University of Louisville. Affiliation: Associated Professor, Universidad Nacional de Colombia.
E-mail: eleonguz@unal.edu.co

[3] Edwin Camilo Cubides Garzón. Mathematician, M. Sc. Applied Mathematics, Ph. D. (c) Computer Science, Universidad Nacional de Colombia, Colombia.
E-mail: eccubidesg@unal.edu.com

[4] Arles Ernesto Rodríguez Portela. Computer Systems Engineer, Universidad Distrital Francisco José de Caldas, Colombia. M. Sc. Computer Systems Engineering, Ph. D. (c) Computer Systems Engineering, Univesidad Nacional de Colombia, Colombia. Affiliation: Assistant professor (BESP program), Universidad Nacional de Colombia, Colombia. E-mail: aerodriguezp@unal.edu.co

[5] Julián Ricardo Maecha D'Maria. Computer Systems Engineer, Universidad Nacional de Colombia, Colombia. Affiliation: Masters student of Computer Systems Engineering, Universidad Nacional de Colombia, Colombia. E-mail: jrmahechad@unal.edu.co

[6] Juan Camilo Rubiano Zambrano. Computer Systems Engineering Student, Universidad Nacional de Colombia, Colombia. Computer Systems Engineering Student, Universidad Nacional Abierta y a Distancia, Colombia. Affiliation: Computer Systems Engineering Student, Universidad Nacional Abierta y a Distancia, Colombia.
E-mail: ing.jcrubianoz@gmail.com

How to cite: Gómez, J., León, E., Cubides, E. C., Rodríguez, A. E., Maecha, J. R., & Rubiano, J. C. (2014). Evolution of teaching and evaluation methodologies: The experience in the computer programming course at the Universidad Nacional de Colombia. *Ingeniería e Investigación*, *34*(2), 85-89.

Introduction

The traditional educational system is built upon a long-established set of customs, which 'guarantees' knowledge transfer from the professor (person who teaches) to the students (persons requiring this knowledge). These customs include: physical meetings in a well-established place (classroom), where a professor can transfer knowledge by talking to his/her passive students (lecture); the linear, built and defined support material (books, class notes, etc.) that is followed by the professor and students in a linear (from start to end) way; and standardized tests used for determining, in a general way, if students are acquiring (memorizing or appropriating) this knowledge in the predefined time periods.

In the last three decades (thanks to computing and communications technologies), educational models have been transformed from monolithic linear models to flexible non-linear models centered on the skills and abilities of each student (Garrison, et al., 2013). This flexible and non-linear model allows students to learn in their own way and at their own pace (guided by several online tools, independent of location (city, region), connection conditions (local or internet) or student time), see Bates (2005), and Garrison, et al. (2013).

In this transformation, several different educational processes, from classroom-based to distance education (using radio, TV, internet and mobile devices), have been included (Zhao, 2010). E-learning is a distance education model that integrates information technologies with pedagogical elements to teach and train "online" students on a particular topic (Bates, 2005, Garrison, et al., 2013, Kearsley, 2000, Salmon, 2004). Moreover, if the content, tests, and tutors are able to adapt according to the abilities of each student, then it is called an adaptive e-learning system (Garcia-Barros, et al., 2005).

According to Gomez et al. (2012), an e-learning system is composed of several subsystems including but not limited to the following: a learning management system for managing, distributing and controlling the training activities; a content management system for defining and maintaining the educational program content (usually course content); an intelligent tutoring system for guiding students in an interactive manner through the educational program content using feedback; a computer adaptive testing (CAT) system for maintaining and applying a test that is able to change the questions according to the ability of each student; and topic interaction and simulation tools, which are comprised of a set of computational elements, such as programming compilers, virtual laboratories, painting tools, and computer-aided design (CAD) engines.

Massive Open Online Courses (MOOCs) are open access courses administered via the web that have a larger student population than a regular course, emphasize interaction among students and do not restrict navigation through the content (Siemens, et al., 2011). This supports the generation of new ideas and viewpoints among students.

A computer programming course has been taught at the UNAL Bogotá Campus for more than 30 years. It is part of the curriculum for all engineering programs. Fourteen years ago, a common methodology for teaching this course was defined. It included material, content, labs, and evaluations. Paper-based exams were taken by all students, and the students' answers were read by a machine and scored. This process took approximately one to two weeks. The course maintained its structure for 10 years with slight changes in the content and programming tools. Four years ago, the evaluation process was improved to an online test, where multiple choice questions were written to compose question pools for each content unit. Pools were uploaded to the Blackboard suite, a different test was generated for each student (from these pools), and the grades were provided to students immediately.

Two years ago, we began to develop an intelligent MOOC platform and to build the content of the computer programming course to test it. Currently, the platform is composed of three systems: a content management system, a learning management system and a testing (adaptive) system. Thus, a course can be defined with a non-linear content unit sructure expressed as a map. Each content unit can be defined as a set of short videos, book chapters, presentations and adaptive tests. In particular, the programming course book was written as a compilation of documents and videos that were recorded in a classroom with students in attendance. The test module of the platform (under validation) includes an adaptive testing approach, i.e., the test adjusts the questions (matching, true-false and multiple choice questions) presented to the student according to his/her answers (ability level).

Additionally, information regarding the user's navigation in the platform is collected. With this information, we expect to apply data mining techniques to analyze the different ways in which students explore the content units and provide an intelligent tutoring system for generating suggestions to improve the learning experiences of the students.

This paper is organized as follows: Section 1 shows the evolution of the programming course methodology. Section 2 presents the e-learning platform and its modules in the content management system, navigation and adaptive evaluation. Finally, the conclusions are drawn.

1. Unifying the Computer Programming Course Methodology: This computer programming course at UNAL can be taken by any undergraduate student (independent of his/her major program), but it is part of the curriculum for all engineering programs. In the last semester term, approximately 900 students took the course (22 groups of 42 students each). Moreover, it was taken by approximately 50 students at three other campuses: Amazonía, Orinoquía and San Andres Islands.

Prior to 2000, the course was offered in a non-standardized way. Each professor taught according to his/her knowledge and programming language preferences. In addition, each content unit was taught by a professor with his/her own personal teaching methodology. Therefore, the content of the session depended on the judgment of the professor (some professors may consider a subject more important than other professors and may therefore emphasize some topics and skip other topics). This variety produces discrepancies in the abilities and skills developed by students, particularly, in computer systems engineering students (these discrepancies were observed in later courses such as data structures and object-oriented programming). Fourteen years ago, a common methodology for teaching this course was defined. It included materials, content, labs, and evaluations. The course was divided into seven units, and each unit was defined with four different sessions: lecture, tutoring, workshop, and computer lab sessions. In the lecture session, a video or professor talks about the unit content. These lecture sessions were repeated several times during the same day (given by different professors), and students were able to attend any of the sessions (according to their time and lecture preferences). The tutoring sessions followed the lecture sessions, usually in the same week, and students were able to interact with their assigned professor and student assistant. These sessions

were designed to allow students to clarify any points of confusion. The workshop sessions followed the tutoring sessions and were designed to test the abilities and skills of the students when they solved a programming problem (from understanding the problem to designing an algorithmic solution for the problem). The computer lab sessions followed the workshop sessions and were designed to determine the technical knowledge of the students (capability of a student for codifying an algorithmic solution in a computer language programming course).

However, following the four consecutive sessions (in this predefined order lecture – tutoring – workshop – computer lab) in all the groups was impossible especially because of holidays (in particular, for courses with classes scheduled on Monday) and occasional non-programmed University closure days because of political and civil protests. Thus, it was impossible to guarantee the same level of knowledge transfer and acquisition in all the groups. As demonstrated, the course used several technological elements but maintained its monolithic linear structure.

Exams were taken by all the students on the same Saturday in a paper-based approach. Because several groups were taught by the same professor, additional professors were required to attend to several groups. Then, the multiple-choice part of the exam was read by a machine, and the open response part of the exam was given to professors for scoring. This process took approximately one to two weeks. Four years ago, the evaluation process was improved to a model of the online joint test. Based on the course material, each professor was asked to define a set of 10 questions (multiple-choice questions) and to define the question pools using a content unit. Pools were uploaded in the Blackboard suite [http://www.campus.virtual.unal.edu.co/], a different test was generated for each student (from these pools) and the grades were provided to students immediately.

To define a more general but standard structure for organizing the content of the programming course, an e-learning platform with a non-linear content unit structure expressed as a map was designed two years ago. Each content unit was defined as a set of short videos, book chapters, presentations and adaptive tests. A book for the programming course was written, and a set of documents, exercises and videos were recorded with the students in traditional class meetings.

2. E-learning Platform

In the definition and implementation of the adaptive e-learning system, several modules were developed to support different courses and manage the course content, classes and users and to evaluate students.

The computer programming course at UNAL defines four roles that were implemented in the e-learning platform: administrator, coordinator, professor and student. There is a unique administrator for the platform, who manages user roles. Coordinators can create as many courses as required and manage their content (which includes a set of resources such as videos, documents and exercises) to help students in their learning process. Additionally, the coordinator can create groups within each course and enroll students and professors in those groups. The coordinator and professors can upload questions for tests and set the parameters for each question for an adaptive evaluation. However, only coordinators can create bags with questions and can set all of the test information and presentation options. Students can access the entire content and can be evaluated in the course and group to which they are registered.

2.1. Navigation

A course is defined as a set of content units. Content units have dependences in terms of time, course organization or content prerequisites. The course is represented in the platform as a directed graph (map) with dependences between the content units to suggest a way for students to navigate the course, e.g., given three topics of the course - Languages, Logic and Sets - we can order topics as follows: Sets depend on Logic, Logic depends on Languages; thus, the graph would be Languages->Logic->Sets (Figure 1 shows the Computer Programming course map). There are two ways to organize the content units: manually, where the selection is performed by the course coordinator, and automatically, where a map is created based on the dependency relationship between the topics. When the map is defined, students have access to the map and navigate over the topics. When a student completes a content unit, the dependencies of this content unit are highlighted for navigation. Thus, students can navigate through these dependencies easily. A central point called the hub allows a student to navigate between all topics in his/her own way.

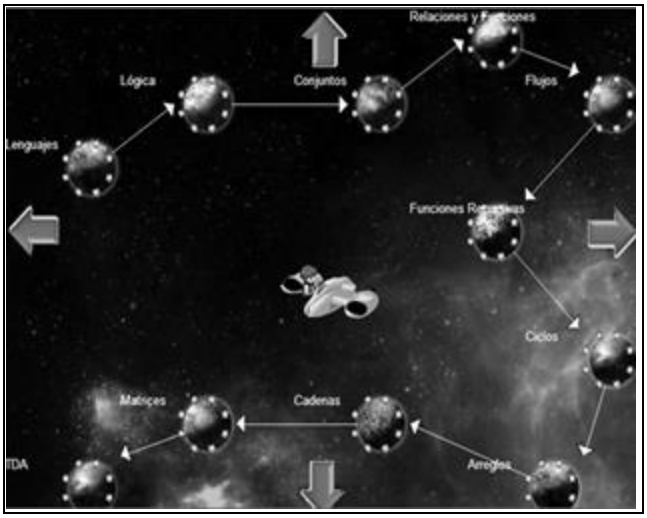

Figure 1. Computer Programming Course Map

2.2. Classes and Content Management

The classes interface is a means of showing the videos and a chat room by topic for sharing information, simulating a classroom and offering a collaborative communication technique. The chat room can be used to send messages to everyone in the class or to a specific user. Classes also have a timeline, which suggests a navigation sequence between the resources of a content unit; however, students are able to navigate videos and resources as they choose. Documents and exercises have their own interfaces.

2.3. Evaluation based on Computer Adaptative Tests

To tackle the problems regarding evaluation (i.e., the delays in calculating grades, the use of the same questions for all students, the huge waste of paper in the evaluation process), we initially used blackboard as an evaluation platform for the computer programming course. Questions were defined as multiple-choice questions, and grades were assigned by the Blackboard platform. However, this model of evaluation has several clear disadvantages. In particular, some tests may be easier than others because some questions can be selected from content units that have different difficulty levels (Blackboard uses a random selection mechanism for selecting the questions regardless of the unit difficulty level).

To solve this problem, a test module that includes computer adaptive tests was developed. A computer adaptive test (CAT), as its name implies, is a software application, where the test content and items adapt to each individual student's ability (Lilley, 2007).

CATs are built upon a group of questions called item banks. Each of the items in the bank has an associated difficulty level parameter (other parameters could be included). Adaptive tests adjust the questions presented to the student according to his/her answers (ability level). Thus, tests are generated based on the answers that students give to a determined question. Questions will also include matching questions, true-false questions and multiple-choice questions with one answer and multiple answers. To create the item banks, the course coordinator sends out a "call for item banks" to the experts (professors of the programming course), and the professors upload questions specifying the difficulty, discrimination and guessing of each question.

To determine whether a student has acquired the concepts and abilities to pass a specific test is a complex task. The professor needs to consider several factors in the test's construction. Generally, there are two extreme situations that professors should avoid: tests that are too easy or too difficult. When the test is too easy, some students may consider it a waste of time, may underestimate the questions and make careless mistakes or may believe that a question is a "trick question". When the test is too difficult, the students could feel discouraged or panic and then just guess at the question (Linacre, 2000). Other factors that professors must consider for test creation are the length and duration of a single test. They need to consider the length (the number of questions) as well as how much time it will take for a student to answer each question to make the test possible.

In our CAT implementation, the item closest to the medium difficulty level is selected as the first question (Lilley, 2007) (Linacre, 2000). To determine the ability of the student (ability estimation) and to select the 'next' question, we use the item response theory (IRT) by considering each item (question) as correct or incorrect (dichotomous). We implement three models of the IRT, which are based on logistic functions: the one parametric logistic (1PL) model, the two parametric logistic (2PL) model, and the three parametric logistic (3PL) model. All these models use a logistic function to establish the probability of answering a question correctly. The main difference between all these models is the set of parameters used in each model: the 1PL model only uses the question's difficulty as a parameter; the 2PL model adds the discrimination factor; and the 3PL model adds the guessing probability as a parameter. To estimate the student's ability, a maximum likelihood function of all the questions answered by the student is used.

To verify when a test is finished, we implemented three options: fixed length, ability estimation and combined options. In the fixed length option, a maximum question number is set, and when a student reaches this number, the test ends. In the ability estimation option, every time a student answers a question, a minimum error value is calculated, and when the ability of the student converges to a fixed value, the student cannot increase or decrease his/her ability in a significant way, and the test ends. In the combined option, we simply use whichever of the options (fixed length or ability estimation) is satisfied first. Both coordinators and professors can upload questions to a course. When they create a new question, it is necessary to specify its type (matching, true-false and multiple choice with one answer or multiple answers), content unit, difficulty, discrimination and guessing probability. The next steps (creation of the questions bank and the test) can only be performed by the coordinator. To create a question bank, the coordinator filters all questions previously uploaded according to criteria, such as the type, difficulty and content unit the questions belong to. Once the questions bank has been created, the final step is to create the test. When the coordinator decides to create an adaptive test, he/she must specify two parameters: the first parameter is the model he/she wants to use for grading the test (1PL, 2PL or 3PL), and the second parameter is the finalization condition, which could be fixed length, ability estimation or combined.

The first trial of the CAT was applied in May 2013. The items repository for the first test had 190 questions uploaded by 12 teachers with each question following the 3PL model parameters. The teacher-made sub-groups reviewed each question's quality; they verified and corrected the questions' content and their parameters.

After taking the test, many students sought an explanation of their grade because they did not understand how it was calculated; they were used to calculations of their grades based on the total number of correct answers over the total number of questions. Teachers had to explain the adaptive test to students and how it was evaluated. Based on the results of the first trial, the model was improved, and several changes were made for the second test. Teachers will have to review the questions more carefully along with the question parameters because those parameters affect the students' grades. We noticed that questions with low difficulty must not have high discrimination values because if a student fails a question with those parameters, his/her grade would be lower.

2.4. Computing Programming Course

This computer programming course is openly accessible and is available to the world in Spanish, but it is specifically offered to all the engineering programs and other programs, such as mathematics and statistics, at the UNAL. We started this project based on the idea of integrating information technologies with pedagogical elements to teach students basic programming. As a result, an e-learning platform was developed with a non-linear content unit structure expressed as a map. Each content unit was defined as a set of short videos, book chapters, presentations and adaptive tests. A book with all the content of the programming course was developed as result of the compilation of documents, exercises and videos that were recorded with students in a classroom meeting.

2.4.1. Book chapters and slides

In 2012, a new methodology was introduced. The lectures were recorded to obtain the main topics of the programming course. From this approach, several book chapters were written and developed as a reference book for the new course. In the old course, students did not have a mathematics foundation to develop the ability to abstract and to propose a mathematician's model to solve programming problems and instead created their programs using a trial-and-error methodology. In the reference book, we do not begin the course with typical programming topics (instructions, conditionals, cycles, arrays, matrices, strings flows, abstract data types), but we first introduce several preliminary concepts regarding a mathematics foundation (logic, sets, relations, functions), and later, we present all the programming concepts as concrete cases of these abstract concepts. The different structures of the programming course are represented as a function and composition of these concepts. Therefore, a program is a composed function that transforms the initial memory into a final memory that contains the solution to the studied problem. From the reference book, we obtained a group of slides for each chapter.

2.4.2. Videos

The first videos that were recorded in the classrooms were used to generate the material for the programming textbook and related slides, but these videos had poor sound quality, and the text on the blackboard was difficult to read. Therefore, we use the slides to generate new videos, where each topic is explained in detail. These videos were generated with a recording program. These videos are short (between three and six minutes), and they highlight the slides. They are clearly legible and have a high quality, which make them easy to read. Additionally, a professor appears in a small box in a corner of the slide and explains the topic of the class, which provides a sense of interactivity with the students and the users.

2.4.3. YouTube Channel

One tool that is very useful in the platform is the YouTube channel of the programming course, which is available online at [http://www.youtube.com/user/PProgramacionUN]. On this channel, we upload the first videos and the old videos recorded in the classroom as the new videos generated with the slides. The platform allows users to watch the videos while the user is browsing the timeline for each unit of content. Additionally, the videos may be watched on the web page of YouTube if the user does not want to enter the platform.

2.4.4. Exercises and Simulation of Tests

For each textbook chapter, a group of exercises was generated from the exercises used in the previous methodology of the course and the exercises proposed by all professors of the course and were used to create the final test each semester. In future semesters, we will provide a simulation test tool to the students so that the students can experiment with the module for the computer adaptive tests prior to the real tests.

Conclusion

This paper has shown the evolution of the programming course at UNAL. From a classic course of programming, this course evolved into a MOOC. This change was made in response to the advancement of computing and network technologies to define a student-centered course and to provide interactivity without time and location restrictions. This course has additional advantages such as an autonomous navigation of the content units and adaptive evaluations to evaluate students according to their abilities. Professors can define question banks to determine the difficulty, discrimination and guessing of each question. This experience is an example of taking a classic course and developing a MOOC based on videos taken in classes with students, converting these videos into content materials for a programming course (book and slides) and finally using the book and the slides to generate better videos and a final structure of the programming course that allows students to explore the course in their order of preference. All stages of the course are evolving while the course is given.

Acknowledgments

This work was supported by the Colciencias Grant No. 110152128971 and the ALIFE and MIDAS research group – Universidad Nacional de Colombia.

References

Bates, A. W. (2005). *Technology, e-learning and distance education* (2nd ed.). Routledge, ISBN 0-415-28436-8.

Garrison, D. R., & Anderson, T. (2013). *E-learning in the 21st century: a framework for research and practice*. Taylor & Francis.

Garcia-Barrios, V. M., Modritscher, F., & Gutl, C. (2005). Personalization versus adaptation a user-centred model approach and its application. In I-KNOW'05 (pp. 120–127).

Gómez, J., León, E., Rodriguez, A., Cubides, C., Mahecha, J., Rubiano, J., & Prado, W. (2012). A Didactic E-learning Platform with Open Content Navigation and Adaptive Exercises. In Proceedings of the ICEELI 2012 Conference (pp. 1-6). IEEE Explore.

Kearsley, G. (2000). *Online education: Learning and teaching in Cyberspace*. Belmont, CA: Wadsworth Publishing Company.

Lilley, M. (2007). *The development and application of computer-adaptive testing in a higher education environment*.

Linacre, J. M. (2000). *Computer-adaptive testing: A methodology whose time has come. MSA memorandum No. 9*. Chicago: MESA psychometric laboratory, University of Chicago.

Salmon, G. (2004). *E-activities: The Key to Teaching and Learning Online* (2nd ed.). London: Taylor & Francis.

Siemens, G., & Downes, S. (2011). *The MOOC Guide*. Retrieved from https://sites.google.com/site/themoocguide.

Zhao, X. (2010). *Adaptive content delivery based on contextual and situational model*. (Ph. D. Thesis). The University of Electro-Communications, Tokyo, Japan.