First International Computer Programming Education Conference

ICPEC 2020, June 25–26, 2020, ESMAD, Vila do Conde, Portugal (Virtual Conference)

Edited by

Ricardo Queirós

Filipe Portela

Mário Pinto

Alberto Simões

Editors

Ricardo Queirós
Politécnico do Porto, Portugal
ricardoqueiros@esmad.ipp.pt

Filipe Portela
Universidade do Minho, Portugal
cfp@dsi.uminho.pt

Mário Pinto
Politécnico do Porto, Portugal
mariopinto@esmad.ipp.pt

Alberto Simões
Instituto Politécnico do Cávado e Ave, Portugal
asimoes@ipca.pt

ACM Classification 2012
Applied computing → Education

ISBN 978-3-95977-153-5

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-153-5.

Publication date
June, 2020

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at https://portal.dnb.de.

Digital Object Identifier: 10.4230/OASIcs.ICPEC.2020.0

ISBN 978-3-95977-153-5 **ISSN 1868-8969** **https://www.dagstuhl.de/oasics**

OASIcs – OpenAccess Series in Informatics

OASIcs aims at a suitable publication venue to publish peer-reviewed collections of papers emerging from a scientific event. OASIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 1868-8969

https://www.dagstuhl.de/oasics

We want to thank all the people involved in the ICPEC conference.

We thank each one of the authors for their valuable contributions.
Our sincere gratitude for their time and expertise to this book of proceedings.

We express our sincere gratitude to the reviewers work,
on improving each and every article.

Finally, a special thank you to the publishing team at OASIcs.

This book is dedicated to our families.

Contents

First International Computer Programming Education Conference (ICPEC 2020).
Editors: Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Preface

At a time when the Covid-19 pandemic is widespread worldwide, many laboratories and research centers are trying to find a solution to the problem of the virus and its mutations. In this difficult period, problem-solving skills are being applied mostly for diagnosing illnesses and developing treatment plans, and, somehow, in the short term, discovering a vaccine.

This is just one example of the transversality of problem-solving skills and its crucial importance at all levels of our society. In reality, problem-solving is one of the key skills of tomorrow's society future. As opposed to a hard skill that is learned mostly through education, problem-solving is nonetheless one of the most valued attributes employers seek in their job candidates. In fact, it's hard to find a professional position that doesn't require problem-solving skills of some kind.

Several levels composed the problem-solving process from analyzing factors which contributes for the problem, generate and evaluate the best solutions, implement a solution to assessing the effectiveness of the implementation. In short, problem solving requires creativity, intuition, knowledge, and skill. Nevertheless, it also requires practice.

Practice in the computer programming domain boils down to solving programming exercises. In the last decades several tools appeared to foster practice by introducing online environments with automatic evaluation. These type of tools relief teachers of the burden of the manual assessment which is clearly time-consuming and error-prone. Despite its regular use, programming courses still have high failure and dropout rates justified by the subject's complexity and obsolete teaching methods. Both affect dramatically the student's motivational levels. In order to overcome this issue, many proposals appeared in recent years to make programming courses more personalized and funnier. Personalization can be obtained through interaction and experience which can be used with machine learning algorithms to adapt the programming exercises to students based on their progression pace and knowledge. Fun has a positive effect on motivation levels, determining what we learn and how much we retain. One of the biggest challenges is how can inject this last facet in existent learning environments. One of he obvious answers is by using gamification. Despite its early success, gamification cannot be seen as the bullet-proof and should be used in a wise and balance way.

It is in this context that educators, scientists and practioners begin to explore new ways to enhance the teaching-learning of problem-solving skills mediated by intelligent online systems with twofold vision: the support of automatic evaluation with rich visual feedback and the delivery of progressive and gamified exercises adapted to different student knowledge levels and profiles.

This book gathers all the accepted articles submitted to the first edition of the International Computer Programming Education Conference (ICPEC). The book presents a comprehensive and recent view of the emerging trends, techniques, paradigms, frameworks and tools for the teaching-learning process in the computer programming domain. At the same time, it identifies new trends on this topic from pedagogical strategies to technological approaches.

Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões

First International Computer Programming Education Conference (ICPEC 2020).
Editors: Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Scientific Committee

Mohammed Abuhelaleh
Al-Hussein Bin Talal University
Ma'an, Jordan

Cristina Alcaraz
Universidad de Málaga
Málaga, Spain

Antonios Andreatos
Hellenic Air Force Academy
Acharnes, Greece

Ana Azevedo
ISCAP/Politécnico do Porto
Matosinhos, Portugal

Daniel Azevedo
Escola Superior de Tecnologia e Gestão de Viseu
Viseu, Portugal

Karolina Baras
Universidade da Madeira
Funchal, Portugal

Jorge Bernardino
ISEC/Instituto Politécnico de Coimbra
Coimbra, Portugal

Bárbara Cleto
Instituto Politécnico do Cávado e do Ave
Barcelos, Portugal

Robertas Damasevicius
Silesian University of Technology
Gliwice, Poland

Micaela Esteves
Instituto Politécnico de Leiria
Leiria, Portugal

Carlo Giannelli
University of Ferrara
Ferrara, Italy

Anabela Gomes
Universidade de Coimbra
Coimbra, Portugal

Pedro Guerreiro
Universidade do Algarve
Faro, Portugal

Sergio Ilarri
University of Zaragoza
Zaragoza, Spain

Manuele Kirsch-Pinheiro
Université Paris 1 Panthéon Sorbonne
Paris, France

Anna Kobusinska
Poznań University of Technology
Poznań, Poland

Zuzana Kubincová
Comenius University
Bratislava, Slovakia

José Paulo Leal
Universidade do Porto
Portugal

António Manso
BioISI: BioSystems & Integrative Sciences Institute
Lisboa, Portugal

Ricardo Martinho
Instituto Politécnico de Leiria
Leiria, Portugal

Silvia Maria Massa
University of Cagliari
Sardinia, Italy

Raffaele Montella
University Parthenope
Naples, Italy

Paula Morais
Universidade Portucalense
Porto, Portugal

Fernando Moreira
Universidade Portucalense
Porto, Portugal

Leonel Morgado
INESC-TEC/Universidade Aberta
Lisboa, Portugal

Alexander Paar
Duale Hochschule Schleswig-Holstein
Lübeck, Germany

First International Computer Programming Education Conference (ICPEC 2020).
Editors: Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

List of Authors

Akiyuki Minamide
International College of Technology
Kanazawa-shi, Ishikawa Japan
minamide@neptune.kanazawa-it.ac.jp

Alae Zouitni
Universität Passau, Passau, Germany
zouitni.alae@gmail.com

Alberto Rojas-Salazar
Trinity College Dublin, Dublin, Ireland
rojassaa@tcd.ie

Alberto Simões
2Ai, School of Technology, IPCA
Barcelos, Portugal
asimoes@ipca.pt

Álvaro Rocha
Faculty of Science and Technology
University of Coimbra, Portugal
amrrocha@gmail.com

Ana Alves
Coimbra Polytechnic – ISEC, Portugal
aalves@isec.pt

Anabela Coelho
Agrupamento de Escolas de Pombal
Pombal, Portugal
anabela.coelho@aepombal.edu.pt

António Vieira de Castro
Department of Informatics, ISEP
Polytechnic of Porto, Portugal
avc@isep.ipp.pt

Bárbara Cleto
Escola Superior de Tecnologia, IPCA
Barcelos, Portugal
a13993@alunos.ipca.pt

Ceres Morais
INESC TEC, Porto, Portugal
Universidade do Estado do
Rio Grande do Norte, Mossoró, Brasil
ceresmorais@uern.br

Cristiana Araújo
Centro ALGORITMI
Universidade do Minho, Braga, Portugal
decristianaaraujo@hotmail.com

Cristina Sylla
Research Centre on Child Studies (CIEC)
Universidade do Minho,Braga, Portugal
cristina.sylla@ie.uminho.pt

Daniela Pedrosa
CIDTFF, Aveiro, Portugal
University of Aveiro, Portugal
dpedrosa@ua.pt

David González-Ortega
University of Valladolid
Valladolid 47011, Spain
davgon@tel.uva.es

Davide R. Carneiro
CIICESI-ESTG, Politécnico do Porto
Algoritmi Center, Department of Informatics
University of Minho, Portugal
dcarneiro@estg.ipp.pt

Diana Barbosa
Centro ALGORITMI
Universidade do Minho, Braga, Portugal
a78679@alunos.uminho.pt

Diogo Ferreira
IOTech – Innovation on Technology, Portugal
University of Minho, Portugal
diogoferreira@iotech.pt

Emanuel Silva
Department of Informatics, ISEP
Polytechnic of Porto, Portugal
ecs@isep.ipp.pt

Fernanda Coutinho
Coimbra Polytechnic – ISEC, Portugal
fermaco@isec.pt

Filip Rezabek
Siemens AG, Munich, Germany
filip.rezabek@siemens.com

First International Computer Programming Education Conference (ICPEC 2020).
Editors: Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Filipe Portela
Algoritmi Research Centre
University of Minho, Portugal
IOTech – Innovation on Technology
cfp@dsi.uminho.pt

Francisco Javier Díaz-Pernas
University of Valladolid
Valladolid 47011, Spain
pacper@tel.uva.es

Frederico Santos
Coimbra Polytechnic – ISEC, Portugal
fred@isec.pt

Glívia Angélica R. Barbosa
CEFET Minas Gerais, Brasil
glivia@cefetmg.br

Gustavo Gutiérrez-Carreón
Universidad Michoacana de
San Nicolás de Hidalgo
Morelia, Michoacán, 58000, México
gagutc@umich.mx

Hirofumi Yamada
Kanazawa Institute of Technology
Hakusan-shi, Ishikawa Japan

Iveta Demková
Comenius University in Bratislava
Leasure-time Center, Šala
Slovakia
ivetkacs@gmail.com

Jakub Swacha
Department of IT in Management
University of Szczecin, Szczecin, Poland
jakub.swacha@usz.edu.pl

Javier Manuel Aguiar-Pérez
University of Valladolid
Valladolid 47011, Spain
javagu@tel.uva.es

J. Pedro Amaro
Coimbra Polytechnic – ISEC, Portugal
amaro@isec.pt

João Cunha
Coimbra Polytechnic – ISEC, Portugal
jcunha@isec.pt

João Durães
Coimbra Polytechnic – ISEC, Portugal
jduraes@isec.pt

João Martinho Moura
Instituto Politécnico do Cávado e do Ave
Barcelos, Portugal
jmoura@ipca.pt

João Sá
Escola Secundária de Avelar Brotero
Coimbra, Portugal
joaosa@gmail.com

Jorge Barreiros
Coimbra Polytechnic – ISEC, Portugal
jmsousa@isec.pt

Jorge Cuellar
Siemens AG, Munich, Germany
Universität Passau, Passau, Germany
jorge.cuellar@siemens.com

Jorge Mendonça
Department of Mathematics, ISEP
Polytechnic of Porto, Portugal
jpm@isep.ipp.pt

José Carlos Paiva
CRACS – INESC-Porto LA, Porto, Portugal
DCC – FCUP, Porto, Portugal
jose.c.paiva@inesctec.pt

José Cravino
CIDTFF, Aveiro, Portugal
University of Trás-os-Montes e Alto Douro
Vila Real, Portugal
jcravino@utad.pt

José Manuel Cerqueira
Instituto Politécnico do Cávado e do Ave
Barcelos, Portugal
cerqueirajm@gmail.com

José Paulo Leal
CRACS – INESC-Porto LA, Porto, Portugal
DCC – FCUP, Porto, Portugal
zp@dcc.fc.up.pt

Kazuya Takemata
International College of Technology
Kanazawa-shi, Ishikawa Japan

Lázaro V. O. Lima
Centro ALGORITMI
Universidade do Minho, Braga, Portugal
lazaro.lima@ifb.edu.br

Leonel Morgado
INESC TEC, Porto, Portugal
Universidade Aberta, Coimbra, Portugal
leonel.morgado@uab.pt

Luís Ferreira
2Ai, School of Technology
IPCA, Barcelos, Portugal
lufer@ipca.pt

Luis Gonzaga Magalhães
Centro ALGORITMI
Universidade do Minho, Braga, Portugal
lmagalhaes@dsi.uminho.pt

Luís M. Alves
Research Centre in Digitalization and
Intelligent Robotics (CeDRI)
Polytechnic Institute of Bragança
Bragança, Portugal
lalves@ipb.pt

Mads Haahr
Trinity College Dublin, Dublin, Ireland
mads.haahr@tcd.ie

Marcela Viana P. Almeida
Research Centre in Digitalization and
Intelligent Robotics (CeDRI)
Polytechnic Institute of Bragança
Bragança, Portugal
CEFET Minas Gerais, Brasil
a42929@alunos.ipb.pt

Marco Silva
Coimbra Polytechnic – ISEC, Portugal
msilva@isec.pt

María Ángeles Pérez-Juárez
University of Valladolid
Valladolid 47011, Spain
mperez@tel.uva.es

Maria João Varanda Pereira
Research Centre in Digitalization and
Intelligent Robotics (CeDRI)
Polytechnic Institute of Bragança
Bragança, Portugal
mjoao@ipb.pt

Maria José Azevedo Brito
Instituto Politécnico de Viana do Castelo
Viana do Castelo, Portugal
Centro de Linguística da Universidade Nova
de Lisboa (CLUNL)
mjazevedo@estg.ipvc.pt

Maria Pinto-Albuquerque
Instituto Universitário de Lisboa (Iscte)
ISTAR, Lisboa, Portugal
maria.albuquerque@iscte-iul.pt

Marílio Cardoso
Department of Informatics, ISEP
Polytechnic of Porto, Portugal
joc@isep.ipp.pt

Mario Madureira Fontes
Universidade Aberta, Coimbra, Portugal
Pontifícia Universidade Católica de
São Paulo, Brasil
mario@mario.pro.br

Mario Martínez-Zarzuela
University of Valladolid
Valladolid 47011, Spain
marmar@tel.uva.es

Mário Pinto
uniMAD, ESMAD, P.PORTO, Portugal
mariopinto@esmad.ipp.pt

Miguel Silva
IOTech – Innovation on Technology, Portugal
University of Porto, Portugal
miguelsilva@iotech.pt

Míriam Antón-Rodríguez
University of Valladolid
Valladolid 47011, Spain
mirant@tel.uva.es

Oksana Yu. Prykhodchenko
Department of Finances
National Metallurgical Academy of Ukraine
Dnipro, Ukraine
oksana.prykhodchenko@gmail.com

Olha S. Shevtsova
Department of Software Engineering
Dnipro University of Technology
Dnipro, Ukraine
shevtsova.o.s@nmu.one

Paula Ramírez-Alfaro
University of Costa Rica
Alajuela, Costa Rica
paula.ramirez@ucr.ac.cr

Paulo F. Gonçalves
Coimbra Polytechnic – ISEC, Portugal
a21171940@isec.pt

Pedro Pinto
Instituto Politécnico de Viana do Castelo
Viana do Castelo, Portugal
INESC TEC, Porto, Portugal
pedropinto@estg.ipvc.pt

Pedro R. Henriques
Centro ALGORITMI
Universidade do Minho, Braga, Portugal
prh@di.uminho.pt

Pedro Vasconcelos
Faculty of Science, University of Porto
Porto, Portugal
LIACC
emailpbv@dcc.fc.up.pt

Raffaele Montella
Università Degli Studi Di Napoli
"Parthenope", Italy
raffaele.montella@uniparthenope.it

Ricardo Queirós
CRACS – INESC-Porto LA
uniMAD – ESMAD, Polytechnic of Porto
Porto, Portugal
ricardoqueiros@esmad.ipp.pt

Rita P. Ribeiro
Faculty of Science, University of Porto
Porto, Portugal
LIAAD-INESC TEC
rpribeiro@dcc.fc.up.pt

Rui Figueiredo
Agrupamento de Escolas Alcaides de Faria
Barcelos, Portugal
ruifigueiredo@aeaf.edu.pt

Rui J. R. Silva
CETRAD|Centre for Transdisciplinary
Development Studies
University of Trás-os-Montes e Alto Douro
Portugal
rui.silva@utad.pt

Salete Teixeira
Centro ALGORITMI
Universidade do Minho, Braga, Portugal
salete.teixeira97@gmail.com

Sara Paiva
Instituto Politécnico de Viana do Castelo
Viana do Castelo, Portugal
sara.paiva@estg.ipvc.pt

Sergii Yu. Semenov
Department of Software Engineering
Dnipro University of Technology
Dnipro, Ukraine
semenovs@gmx.com

Serhii D. Prykhodchenko
Department of Software Engineering
Dnipro University of Technology
Dnipro, Ukraine
prykhodchenko.s.d@nmu.one

Sokol Kosta
Aalborg Universitet, Denmark
sok@es.aau.dk

Teresa Terroso
uniMAD, ESMAD, P.PORTO, Portugal
teresaterroso@esmad.ipp.pt

Tiago Gasiba
Siemens AG, Munich, Germany
Universität der Bundeswehr München
Munich, Germany
tiago.gasiba@siemens.com

Ulrike Lechner
Universität der Bundeswehr München
Munich, Germany
ulrike.lechner@unibw.de

Vitor Manuel Ferreira
Instituto Politécnico de Viana do Castelo
Viana do Castelo, Portugal
ferreira@estg.ipvc.pt

Zuzana Kubincová
Comenius University in Bratislava, Slovakia
kubincova@fmph.uniba.sk

EasyCoding – Methodology to Support Programming Learning

Marcela Viana P. Almeida
Research Centre in Digitalization and Intelligent Robotics (CeDRI),
Polytechnic Institute of Bragança, Portugal
CEFET Minas Gerais, Brasil
a42929@alunos.ipb.pt

Luís M. Alves[1]
Research Centre in Digitalization and Intelligent Robotics (CeDRI),
Polytechnic Institute of Bragança, Portugal
lalves@ipb.pt

Maria João Varanda Pereira
Research Centre in Digitalization and Intelligent Robotics (CeDRI),
Polytechnic Institute of Bragança, Portugal
mjoao@ipb.pt

Glívia Angélica R. Barbosa
CEFET Minas Gerais, Brasil
glivia@cefetmg.br

Abstract

Knowing that the programming curricular units in the first year of engineering courses have a high failure rate and, assuming that this failure is due, in large part, to the lack of motivation and the lack of autonomy of the student to program in context outside the classroom, a methodology based on activity guides using attractive web platforms is proposed. The proposed methodology aims to facilitate both the planning of activities by the teachers and the autonomy and motivation by students. In order to receive a first feedback about this work, the methodology is being used by programming professors from Polytechnic Institute of Bragança, but in the near future it will be also evaluated by professors from the Federal Center of Technological Education of Minas Gerais and from the Federal Technological University of Paraná, both from Brazil. Following this work, a system is being developed that allows the automatic construction of guides based on exercises available from the web and systems that facilitate the collection of solutions and analysis of results.

2012 ACM Subject Classification Human-centered computing → Visualization systems and tools; Software and its engineering → Imperative languages; Social and professional topics → Computer science education

Keywords and phrases learning programming, teaching programming, automatic activity guides, programming motivation

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.1

Funding This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the Project Scope: UIDB/05757/2020.

1 Introduction

School dropout and failure in higher education has been the subject of many research studies. Student dropout is an international problem that has social consequences and affect the results of educational institutions [1]. This is due to several factors, such as the lack of

[1] Corresponding author

motivation and dedication of students in the face of the challenges that are proposed to them inside and outside the classroom. Following this context, Mendes [14] quotes that the first programming curricular units at the Department of Informatics Engineering from the University of Coimbra suffer from high failure and dropout rates, as reported in many other high educational institutions. As Silva Filho [18] quotes, in Brazil, the private sector invests around 2% to 6% of higher education institutions' revenues in marketing, in order to attract new students.

However, in order to keep students enrolled, just short and punctual experiments have been made in higher education contexts [17, 13, 16]. Moreover, developing a programming logic can be a complex and challenging task. This is because, according to Moreno [15] and França et al. [8], in addition to knowing the basic instructions, the individual needs to understand how to use them to solve different problems. Furthermore, the applicability of this content in other areas has expanded the profiles of people interested in learning it (e.g., high school students). Therefore, it is necessary to diversify the approaches to teaching/learning programming so that they are suitable for different profiles of interest [8, 15]. Regarding this issue, given the constant unsatisfactory results in the first programming disciplines of technology courses in Portugal and Brazil, a larger study on new methodologies to support programming learning become essential in the search for better teaching quality and greater enrollment of students.

Following the results analysis of the project NoviBraga [4], the objective of this study is to avoid the lack of dedication of the first year programming students. To this end, teaching methodologies are being developed to improve the quantity and the quality of the slots of time that the student uses to program outside and inside classroom and consequently improve the learning outcomes. According to Gomes and Mendes [9], when talking with programming teachers, most of them claim that students don't know how to program because they don't know how to solve problems and don't have enough mathematical background. There are lots of tools on the web that can be used by programming students and almost all of them have some motivating features like animations, control flow graphs, interactive debugging, syntax-direct editors, collaborative programming, chats, programs to complete, games, submission platforms, rankings and so on. All these features can turn the programming activity more attractive and easier if they are correctly explored. Moreover, these tools can be combined to join the best of each one. In order to achieve this, teachers also need some support to be able to propose the appropriate set of activities along the semester and get feedback from the students. An activity guide is needed, and the exercises must be carefully chosen. It is difficult to find in the web these activity guides already constructed. The main contribution of this work is to propose a methodology based on web platform activities and to create a system to automatically produce activity guides. These guides are based on free web platforms whose features we believe that can motivate students to program. This work presents two research questions:

1. Does the use of web systems that allow the resolution of programming exercises motivate students to work outside the classroom? (Section 3)
2. Is it possible to systematize and automate the creation of programming activity guides for both the teacher and the student? (Section 4)

Besides this introduction where some related works are presented, the section 2 introduces the methodology proposed. The application of the activity guides is discussed in section 3 and based on this experience, a system to automatically generate those guides is presented in section 4. In section 5, some conclusions will be presented as also some clues for future work.

2 Methodology

In order to follow our approach, the following steps are proposed:

1. Collection of information on C programming web teaching platforms;
2. Creation of a classification system for web platforms, with the objective of using them inside and outside the classroom context;
3. Classification of platforms found and creation of a website to make this study available to C programming teachers;
4. Creation of activity guides to be developed during the first programming semesters with the students, with respective monitoring;
5. Validation of the methodology, based on the results of the monitoring, the students' academic success and questionnaires for collecting opinions from teachers and students;
6. Development of a system that allows teachers to assemble and generate new activity guides, according to characteristics pre-defined by the classification model.

The result of the three first items can be seen in [3]. At the end of the project's development, the results will be analyzed and validated, and future perspectives and proposals for improvement of the study will be pointed out.

3 Application of the Activity Guides

The present study considered an introductory programming course (Programming I) in the context of higher education at Polytechnic Institute of Bragança (IPB). This course unit has 218 enrolled students spread over two courses, namely, Informatics Engineering and Management Informatics. Programming I is in the first year, first semester of our two courses. At the end of these course unit, the student is expected to be able to design solutions and implement C programs that solve small/medium complexity problems. In order to do that, the student must apply concepts of imperative programming in the C programming language, coding function-based structured programs to manipulate data structures. The student must also be able to use an Integrated Development Environment (IDE) including the debugging tool to get successful solutions. In this case, the IDE is the Microsoft Visual Studio Community 2019. The syllabus of this course contains C language topics that are commonly taught in introductory programming at the university level. Like other cases found in the literature, Programming I also reveals a low student success rate. The teaching-learning programming process is a difficult task, as is known to the community involved. This process faces several challenges, the most important of them, it is to improve student motivation to learn and to ensure that they do not give up learning. Another important challenge is to enroll the students to work outside of the classroom. Aware of these challenges, in the first semester of academic year 2019/2020 we applied the proposed approach to hold the students involved in programming tasks. Thus, we create a set of activity guides that cover the whole syllabus of programming I. These guides use a set of web platforms with exercises well contextualized. In the classes, the teacher gave a brief description of each of these guides, whose exercises were solved by the students inside and outside of the classroom. In a first phase, we proposed the activity guides to 65 students that are divided into two different classes. We decided to involve only these students for two reasons:

(i) The teacher, one of the authors of this paper, taught both classes, so he had full control of the students;
(ii) The idea is to start just as a pilot project and adjust the guides during the semester.

The Figure 1 presents the results of the activity guides application. We enumerated the activity guide from 1 to 5 and they are divided by topic. For each topic we have more than one activity guide. The table shows also the web platforms used by the students to do the tasks. A qualitative research was carried out on the main tools available online, which support teaching C programming, focusing mainly on features considered useful for combating school dropout. The chosen platforms are URI Online Judge, Codeboard, CPuzzles, C Tutor, CodinGame and Coderbyte [11, 5, 12, 10, 7, 6]. They were chosen based on their characteristics like quantity of exercises, contents scope, ease of use, free access and some extra features like tutorials, animation, collaborative work and automatic assessment. All these web platforms have different purposes and they cover a set of different functionalities:

1. URI Online Judge is a web portal that offers a variety of programming problems for the students to solve and share knowledge, besides having a teachers' module called URI Online Judge Academic that lets the teacher manage the activities, students' submissions and grades, without having to use any other system. Its main features include real-time correction, available problems separated by categories and use of gamification concepts, with the application of a of a *badges* and *ranks* reward system.
2. Codeboard is a web-based IDE to help the students to learn programming by allowing the teachers to create exercises as they wish and letting the students run and compile their own solutions inside the IDE and after submitting to the teacher, that has total access to the student answers. Codeboard also allows automatic grading to help the exercises correction and has the possibility of exporting the results in spreadsheet format.
3. CPuzzles is a repository that contains a collection of C puzzles and solutions to supplement the activities and assist the teacher. The puzzles inside the platform are divided in eight groups, each one regarding a C subject. Each puzzle has a difficulty rating, in order to help the teacher choose which exercise better meets students needs. Besides that, the teacher can find an example of solution and a *"code skeleton"*, which helps the student to start the exercise.
4. C Tutor is a tool that assists teachers and students when writing code, making them understand what happens as the computer runs each line of code and visualize each state of data structures being used. When applicable, C Tutor also helps the visualization of the contents present in the heap and the pointers. Finally, the system allows collaborative work, by letting different users change the same code simultaneously and has an online chat that allows students to discuss each problem.
5. CodinGame is a gamified platform for teaching programming languages. The system uses game puzzles with high level animation and different difficulty levels. The website has its own IDE where the students can run and compile their own code and submit their result to the platform that has its own automatic correction. CodinGame also has a reward system with the application of *badges* and *ranks*. There is still a forum dedicated to the users for any doubts about the puzzles.
6. Coderbyte is a web system developed to offer challenges and courses with the aim to help users to prepare for upcoming job interviews, as well as practicing more programming. The system has an IDE where the students can run, compile and test if their solution is valid, by going through the system's automatic correction. The code challenges range in difficulty and can be solved using different programming languages.

In Figure 1 we can see the code and the name of each exercise made by the students. These codes were obtained by the platform itself. In the last three columns of Table 1 we can see the number of total students to whom the activities were proposed, the number of student submissions and the number of successful submissions. Based on these numbers we quickly

Activity Guide	Web Platform(s)	Exercises	Number of Students	Number of Student Submissions	Number of Successful Submissions
Elementary data types					
1.0	URI Online Judge and Codeboard	1002 – Circle Area 1005 – Average 1 1006 – Average 2	65 65 65	27 24 23	24 16 16
Testing and conditions					
2.0	URI Online Judge	1041 – Coordinates of a Point 1042 – Simple Sort	65 65	22 22	18 19
2.1	URI Online Judge	1044 – Multiples 1046 – Game Time	65 65	22 19	17 15
Loops					
3.0	CPuzzles and C Tutor	Puzzle A03 Puzzle A10	65 65	24 24	24 24
3.1	URI Online Judge	1060 – Positive Numbers 1064 – Positives and Average	65 65	10 10	10 6
3.2	CodinGame	Classic Puzzle – Temperatures	65	3	3
3.3	URI Online Judge and Codeboard	1071 – Sum of Consecutive Odd 1132 – Multiples of 13	65 65	9 8	8 7
Functions					
4.0	Coderbyte	Time Convert First Factorial	65 65	9 9	9 9
4.1	CodinGame	Classic Puzzle – The River I	65	1	1
Arrays					
5.0	CPuzzles and C Tutor	Puzzle C01 Puzzle C03	65 65	2 2	1 1
5.1	URI Online Judge	1173 – Fill the Vector I 1174 – Vector Selection I	65 65	2 2	1 0
5.2	CodinGame	Classic Puzzle – The Descent	65	2	2

Figure 1 Results of Activity Guides Application.

realized that the student involvement is much greater in the first activities and it has been decreasing as the subjects have become more complex. The deadline for students to complete the tasks of the last three activity guides is still open, so it is expected that more students will complete them. The Coderbyte and CodinGame web platforms had less adherence by students. In fact, some of the students referred some difficulties to use and to understand the problem statement of these platforms. All the activity guides were transferred to the students using Sakai platform. This is a web platform similar to the Moodle used in IPB. During the classes, the teacher briefly explained the tasks for each of the activity guides. The students received the activity guides by institutional email. The text of the email always referred some encouragement words. Unfortunately, not all students read institutional email, especially those who miss classes. We need to address this issue during the next semester and use a different approach to convince students to do the activities.

After each activity the teacher opinion was collected and used to improve the next guides. The idea now is to intensify the construction of new activity guides using the platforms that had more success from the students' point of view and also the ones that are more appropriate for the contents of Programming II. Since data structures will be better explored, we are aware that animation tools will be more adequate and useful.

4 A System to Automate the Activity Guides Construction

A system (Figure 2) called EasyCoding is being developed to automate the guides construction and help teachers to produce new activity guides in order to use them with the students inside and/or outside the classroom, with the aim of facilitating their planning and to motivate the students by performing these exercises. The technologies being used for the development of this platform are Javascript with JQuery library to simplify the scripts interpreted on the client-side browser, jsPDF library which is a widely used solution for client-side PDF generation, in order to generate the activity guides for teachers and students, and Bootstrap front-end library to help prototype the system for a friendly and responsive experience. As a

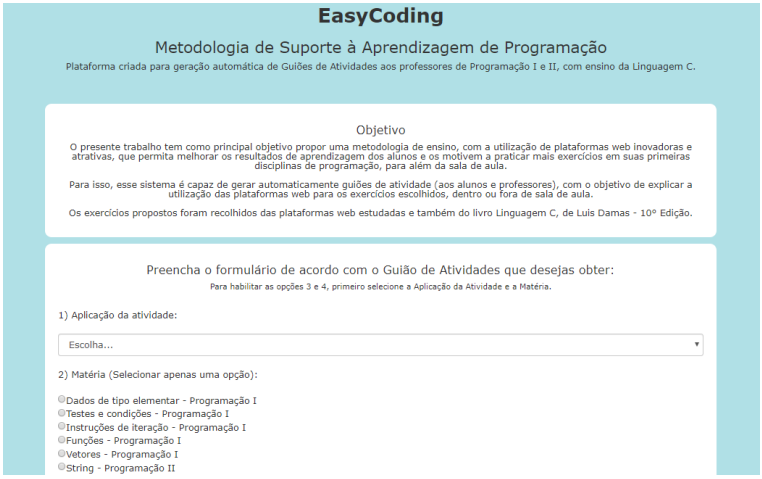


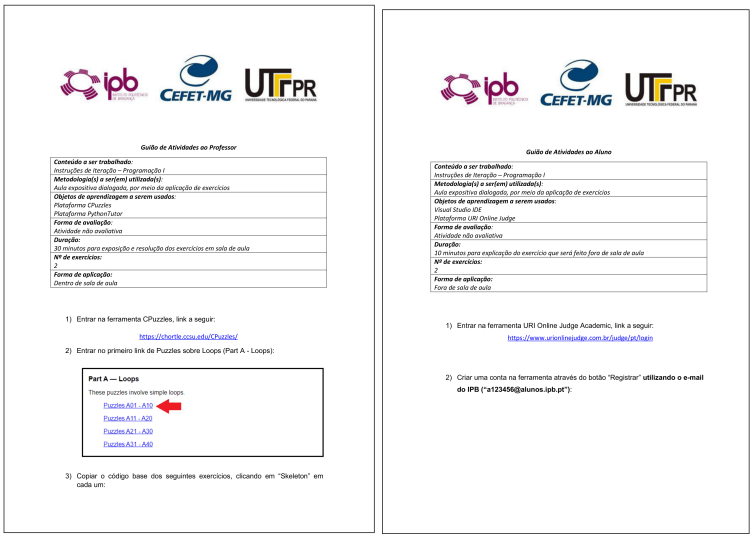
Figure 2 EasyCoding - automatic generation of programming activity guides.

static platform, the system is being hosted on GitHub Pages host service. Therefore, the system is capable of automatically generating activity guides, in order to explain to students and teachers the correct use of each chosen web platform. The proposed exercises were collected from the studied web platforms (when they had exercises available as a feature) and also from the book "Linguagem C", by Luis Damas. When using the system, the teacher only has to choose:

1. Inside or outside the classroom context.
2. Programming subject.
3. Platform in which s/he wants to apply the exercises.
4. The exercises themselves.

Each exercise will have an estimated time in which the teacher can rely on to compose her/his activity guide. Actually two guides will be produced, one for the teacher to prepare the session and one for the student, that will receive it from the teacher, to follow the guide. The results will be automatically collected, and the teacher can use them as s/he wants. The system proposed is already available in [2].

The main reason for the generation of these two types of activity guides (teacher and student guide) is that each web platform being used has its own way of use. Some of them, like URI Online Judge and Codeboard have different functionalities for teachers and students. Because of this, each guide is developed to help their understanding of the steps they have to complete to accomplish the task. Thus, the teacher's guide will assist the teacher on how to use the web platform chosen to add the activities needed and show them to the students. The students' guide will assist them on how to have access to the web platform, perform the given activities and send their results to the teacher. In order to evaluate the acceptance of the analyzed web platforms, an interview was carried out with the students who participated in this study. The interview revealed that the URI Online Judge, Codeboard and Python Tutor platforms had a greater acceptance among the students. According to them, the other ones seemed complex or without significant benefits. Due to this result, we opted to keep on working, this next semester, only with these 3 web platforms and CPuzzles, to collect exercises. Moreover, the EasyCoding system is being implemented to provide activity guides using these 4 platforms also. Examples of generated teacher and student's guides are shown in Figure 3.



Guião de Atividades ao Professor

Conteúdo a ser trabalhado: *Instruções de Iteração – Programação I*
Metodologia(s) a ser(em) utilizada(s): *Aula expositiva dialogada, por meio da aplicação de exercícios*
Objetos de aprendizagem a serem usados: *Plataforma CPuzzles* *Plataforma PythonTutor*
Forma de avaliação: *Atividade não avaliativa*
Duração: *30 minutos para exposição e resolução dos exercícios em sala de aula*
Nº de exercícios: *2*
Forma de aplicação: *Dentro de sala de aula*

1) Entrar na ferramenta CPuzzles, link a seguir:

https://chortle.ccsu.edu/CPuzzles/

2) Entrar no primeiro link de Puzzles sobre Loops (Part A - Loops):

Part A — Loops

These puzzles involve simple loops

Puzzles A01 - A10

Puzzles A11 - A20

Puzzles A21 - A30

Puzzles A31 - A40

3) Copiar o código base dos seguintes exercícios, clicando em "Skeleton" em cada um:

Guião de Atividades ao Aluno

Conteúdo a ser trabalhado: *Instruções de Iteração – Programação I*
Metodologia(s) a ser(em) utilizada(s): *Aula expositiva dialogada, por meio da aplicação de exercícios*
Objetos de aprendizagem a serem usados: *Visual Studio IDE* *Plataforma URI Online Judge*
Forma de avaliação: *Atividade não avaliativa*
Duração: *10 minutos para explicação do exercício que será feito fora de sala de aula*
Nº de exercícios: *2*
Forma de aplicação: *Fora de sala de aula*

1) Entrar na ferramenta URI Online Judge Academic, link a seguir:

https://www.urionlinejudge.com.br/judge/pt/login

2) Criar uma conta na ferramenta através do botão "Registrar" **utilizando o e-mail do IPB ("a123456@alunos.ipb.pt")**:

Figure 3 First page of a teacher and a student's activity guide applied.

5 Conclusion and Future Work

In this paper, a methodology based on the creation of programming activity guides was described. The concrete application of the proposed approach to IPB students was explained as well and the results from this initial research were analyzed. From the first semester of application, we are able to conclude which are the most appropriate tools and what kind of activity guides should be produced to enrich our system and improve the student's motivation. The student opinion about this was also collected and used to improve the guides. The developed system has a database of activities that will be used by the teachers to automatically generate more guides as they wish. At the end of this second semester, the success rate of the Programming I and Programming II subjects will be analyzed and this study will be able to show if the application of these activity guides with the use of web platforms that have motivating features was satisfactory. For that, student and teacher opinions will be collected again.

References

1 Leandro S Almeida and Rosa Vasconcelos. Ensino superior em portugal: Décadas de profundas exigências e transformações. *Innovacion Educativa*, 2008.

2 Marcela Almeida. Easycoding – metodologia de suporte à aprendizagem de programação. `https://marciviana.github.io/projeto_mestrado.html`. Accessed: 2020-04-01.

3 Marcela Almeida. Metodologia de suporte à aprendizagem de programação. `https://mestrado-marcelaviana.webnode.com/`. Accessed: 2020-04-01.

4 Luís Alves, Dušan Gajić, Pedro Rangel Henriques, Vladimir Ivančević, Maksim Lalić, Ivan Lukovic, Maria João Pereira, Srđan Popov, and Paula Tavares. Student entrance knowledge, expectations, and motivation within introductory programming courses in portugal and serbia. *47th European Society for Engineering Education (SEFI 2019)*, pages 1354–1363, 2019.

5 Codeboard. Codeboard – the ide for the classroom. `https://codeboard.io/`. Accessed: 2020-04-01.

6 Coderbyte. Code screening, challenges & interview preparation. `https://coderbyte.com/`. Accessed: 2020-04-01.

7 CodinGame. Play with programming - codingame. `https://www.codingame.com/`. Accessed: 2020-04-01.

8 RS d França, VFS Ferreira, LCF de Almeida, and HJC do Amaral. A disseminação do pensamento computacional na educação básica: lições aprendidas com experiências de licenciandos em computação. In *Anais do XXII Workshop sobre Educação em Computação (WEI-CSBC)*. sn, 2014.

9 Anabela Jesus Gomes and António José Mendes. A study on student performance in first year cs courses. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*, pages 113–117, 2010.

10 Philip Guo. Python tutor – visualize code and get live help. `http://pythontutor.com/`. Accessed: 2020-04-01.

11 Universidade Regional Integrada. Uri online judge. `https://www.urionlinejudge.com.br/`. Accessed: 2020-04-01.

12 Bradley Kjell. Cpuzzles. `https://chortle.ccsu.edu/CPuzzles/CPuzzlesMain.html`. Accessed: 2020-04-01.

13 Christian Maekawa, Walter Nagai, and Claudia Izeki. Relato de gamificação da disciplina projeto e análise de algoritmos do curso de engenharia de computação. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 4, page 1425, 2015.

14 António José Mendes, Luis Paquete, Amilcar Cardoso, and Anabela Gomes. Increasing student commitment in introductory programming learning. In *2012 Frontiers in Education Conference Proceedings*, pages 1–6. IEEE, 2012.

15 Julián Moreno. Digital competition game to improve programming skills. *Journal of Educational Technology & Society*, 15(3):288–297, 2012.

16 Walter Nagai, Claudia Izeki, and Rodrigo Dias. Experiência no uso de ferramentas online gamificadas na introdução à programação de computadores. In *Anais do Workshop de Informática na Escola*, volume 22, page 301, 2016.

17 Walter Aoiama Nagai and Claudia Akemi Izeki. Relato de experiência com metodologia ativa de aprendizagem em uma disciplina de programação básica com ingressantes dos cursos de engenharia da computação, engenharia de controle e automação e engenharia elétrica. *Revista de Exatas e TECnológicas*, 4(1):18–27, 2013.

18 Roberto Leal Lobo Silva Filho, Paulo Roberto Motejunas, Oscar Hipólito, and Maria Beatriz de Carvalho Melo Lobo. A evasão no ensino superior brasileiro. *Cadernos de pesquisa*, 37(132):641–659, 2007.

Challenges and Solutions from an Embedded Programming Bootcamp

J. Pedro Amaro
Coimbra Polytechnic – ISEC, Portugal
amaro@isec.pt

Jorge Barreiros
Coimbra Polytechnic – ISEC, Portugal
jmsousa@isec.pt

Fernanda Coutinho
Coimbra Polytechnic – ISEC, Portugal
fermaco@isec.pt

João Durães
Coimbra Polytechnic – ISEC, Portugal
jduraes@isec.pt

Frederico Santos
Coimbra Polytechnic – ISEC, Portugal
fred@isec.pt

Ana Alves
Coimbra Polytechnic – ISEC, Portugal
aalves@isec.pt

Marco Silva
Coimbra Polytechnic – ISEC, Portugal
msilva@isec.pt

João Cunha
Coimbra Polytechnic – ISEC, Portugal
jcunha@isec.pt

Abstract

Due to the proliferation of IT companies developing web and mobile applications, computer programmers are in such high demand that universities can't satisfy it with newly graduated students. In response, some organisations started to create coding bootcamps, providing intensive full-time courses focused on unemployed people or individuals seeking for a career change. There is, however, a different set of skills that is becoming increasingly required, but is not addressed by those courses: embedded programming. In fact, the Internet of Things is connecting every device to the internet, thus making knowledge on hardware and C/C++ programming very relevant skills.

A group of computer science and electrical engineering university teachers, in collaboration with several industry stakeholders, have promoted an embedded systems programming course in C and C++. This course is based on an intensive project-based approach comprising 6 months of daylong classes followed by 9 months of paid internships. After two editions, thirty embedded programmers, with no relevant previous programming experience, have been placed with the partners' working force.

In this paper, the course organisation and pedagogical methodologies are described. Problems, challenges and adopted solutions are presented and analysed. We conclude that in spite of the intense rhythm and demanding nature of the subject matter, it is possible to find the structure and solutions that keep students engaged and motivated throughout the course, allowing them to gain the required competences and successfully transition into a new career path.

2012 ACM Subject Classification Social and professional topics → Computing education

Keywords and phrases Coding Bootcamp, Embedded Programming, Career Change

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.2

1 Introduction

The increasing demand for specialised workforce in the Information Technologies (IT) industry, due to the digital transformation process that has been gradually taking place in recent years, has resulted in the shortage of highly skilled professionals. This led to the proliferation of short courses and bootcamps focusing in IT skill development, with varying degrees of success [11, 3, 10]. These courses are often concerned with web, mobile or desktop development, whereas embedded systems programming is not usually addressed. In fact, this area requires highly specialised hardware and software development skills, which can be challenging to

acquire in the typically short time frame of these courses. However, there is a growing demand for this type of professionals, due to factors such as the digitalisation of the automotive industry and the Internet of Things (IoT). To meet this demand, a professional re-qualification course, the *Apostar em TI* (AeT) programme [1], was jointly created by a higher education (HE) institution (Coimbra Polytechnic - ISEC) and several industrial partners. The course is highly focused on the skills and competences identified as priority by the industrial partners, and is targeted at highly motivated, full-time students with previous HE experience (i.e., have at least been enrolled in an undergraduate programme). No prior IT background is required or expected.

Having now successfully completed two editions of the course, some key issues were identified and addressed. In this paper, we describe the challenges both students and staff faced, some solutions that have been applied and lessons learned, dealing with the intense and demanding rhythm, short time frame, diverse student's background and qualifications, and the relatively steep learning curve of the subject matter.

The remainder of this paper is structured as follows: Section 2 provides a description of the course. Section 3 describes the results of both editions, lessons learned and adaptations made when transitioning from the 1st to the 2nd edition, and employer feedback. Section 4 presents the main conclusions.

2 The Apostar em TI programme

The *Apostar em TI* (AeT) programme addresses two areas of expertise: Programming (C language [4], with some notions of Software Life Cycle [8] and C++ [9]) and Embedded Systems [5] (Digital Systems, Computer Architecture and Organization, Interfaces and Communication, Real-Time Systems). The complementary internship training guarantees that students achieve a full integration with industrial environment and practices required by the partner companies. In the next sections the AeT programme is presented.

2.1 Goals of AeT

The global market for embedded systems has evolved considerably in recent years. This includes the technology and the industries served. With the advent of IoT and Industrial IoT (IIoT), embedded systems technology has become a central facilitator for the rapidly expanding industries dealing with smart ubiquitous devices and IoT.

Professionals specialised in embedded systems are currently in high demand and most of the regional companies in this sector are actively looking for those professionals, and in many cases are unable to fulfil their needs. In fact, this scenario has specifically been pointed out by our industrial partners and is a key motivation factor for their participation in this project. On the other hand, many other areas are experiencing a decline in demand due to automation and changes in the global society aspects, e.g., Civil Engineering and Chemical Engineering. This creates a large body of professionals looking for an opportunity to change career and for courses that enable them to obtain the required new skills. Computer related engineering is currently one of the most active areas and it is no surprise that professionals from other areas are looking to change specifically to computer related professions.

In this context, one of the main goals of AeT is to offer an opportunity to candidates looking to change their careers to the IT industry. AeT offers the opportunity to acquire the required skills and support for initial placement in the industrial partners. The other goal is to meet the industry needs for IT experts by providing new professionals with the skills in demand.

2.2 Learning outcomes

This course aims at training students into understanding C and C++ programming concepts as well as Embedded Systems usage and implementation mechanisms. Moreover, it aims at developing social and working skills in students, such as teamwork, resilience, communication, time management or responsibility.

2.2.1 Programming/embedded

At the end of the course students should be able to program a microcontroller with a number of peripherals.

Students must learn how to program in C language, including all the normal language structures, constructs and libraries, and also low-level operations, such as the manipulation of bits and registers, which imply the notions of digital numbering and encoding.

For Embedded Systems, students should be able to program both simple microcontrollers, such as 8051s, and more complex ones, such as STM32 ARM architectures. They should also understand and use a number of peripherals such as Accelerometers, ADCs, Led Arrays, among others.

As special requests from the industrial partners, students should also understand the object-oriented paradigm and be able to write simple programs in C++. They should also understand the basic principles of real-time programming, as well as some software engineering practices, including software testing and version control.

2.2.2 Soft skills

The development of soft skills is highly required by the industry, so this programme was organised in a way where students could find an environment similar to a company. So the school assigned one classroom for exclusive use of this course, meaning that students can stay all day long in classes and are free to manage they schedule and study time, which contributes to improve their self-management soft skills. It also encourages a significant shared environment allowing multiple students to remain on their own for extended periods of time, promoting collaboration and competition.

2.3 Structure

The AeT programme is developed in two phases. The first is a lesson-based period where students learn programming in C and C++, embedded systems using 8051 [6] and the ARM-based STM32 [7], and real-time operating systems [2]. The second part consists on an internship in one of the industrial partners.

1. Academic Phase held at ISEC's premises:
   - duration of twenty weeks, between February and July;
   - lecture of 200 hours of theoretical-practical classes by ISEC teachers;
   - 300 hour tutorial training by teachers and instructors;
   - presentation of workshops from partner companies and other guests;
   - execution of a 3 weeks final embedding system project.
2. Professional Internship Phase, to be held in one partner company:
   - duration of 9 months, between September and June;
   - paid professional internship;
   - supervision by ISEC teachers;
   - intermediate and final presentations at ISEC for all participants on the programme about their internship ongoing work and final results.

This course does not confers a degree. However, students who successfully complete the academic phase will be awarded a diploma by ISEC.

2.4 Pedagogical methodology

The pedagogical approach is essentially based on practical training, with exposition based on examples and case studies, and on daily practical work. During the academic phase, students have a weekly average of 12 hours of classes and 15 hours of tutorial support. The expected workload from students is 36 hours per week, summing a total of 720 hours of effort, corresponding to 28 ECTS credits.

The lessons and scheduling of the subjects of the course were carefully thought and planned, with the close participation of the partner companies. It was decided to have two subjects being taught at the same time, meaning lessons related with Programming and Embedded Systems were interleaved during the week. This was aimed at preventing excessive impermeability across topics. Interleaving two topics would allow students to better relate them and think how one could be used with the other.

Each day was organised into three parts:

- During the **morning** the lesson was basically theoretical, but always supported with practical examples and demonstrations.
- In the **beginning of the afternoon**, the first two hours (it could vary from one day to the other) was devoted to exercises for practice.
- **Later in the afternoon** exercises for grading were given to the students.

The students were accompanied by one instructor that was present every afternoon, to clarify doubts and help solving impediments. Instructors were recruited PhD students and professionals with proven high skills in Programming and Embedded Systems.

The evaluation of the students was essentially continuous, based on the quality of the work developed and presented. Each assignment has a set of deliverables that were submitted to GitHub and immediately evaluated, promoting continuous improvement. To encourage engagement, attendance of the students was measured and taken into account for evaluating purposes.

Each student was assigned one teacher as a tutor. This tutor would be responsible for a more direct contact with his assigned students, making sure that their specific difficulties with the subjects, or other individual pedagogical requirements were listened to and taken care of.

All necessary devices, instruments and bibliography are provided by ISEC, however, each student was encouraged to have his own laptop.

2.5 Recruitment

The process of recruitment involved one of our industrial partners and the faculty. The industrial partner supplied the know-how of its human resources office, interviewing candidates using their own psychological and psychometric tests. The faculty also interviewed each candidate to understand their motivations and assess their logical thinking abilities. These interviews always had one teacher and the psychologist from the industrial partner. The results from both parties were confronted towards a combined opinion. The evaluation resulting from each interview were later analysed in a meeting involving all the faculty and the outcome was the result of a consensus. In the second edition, a supplementary programming test was also created to decide on the cases where the faculty was not sure on acceptance or rejection.

Table 1 Profile of initial candidates to AeT.

	Academic degree			Previous studies in STEM?		Gender		Age		
	no	Bac	MSc	Yes	No	M	F	< 30	< 40	≥ 40
Edition 1	19%	55%	26%	54%	46%	78%	22%	43%	46%	11%
Edition 2	20%	65%	15%	51%	49%	80%	20%	34%	40%	26%

3 Outcomes and discussion

The first edition of AeT started in February 2018, and the second edition in February 2019. At this point the first edition of the programme is complete, and the second edition ended the academic phase. This allows us to extract useful information about the recruitment process, lessons' management and students' performance, obtaining some insights for future editions.

3.1 Candidates and recruitment process analysis

Despite the short marketing and application period (about 2 months), 162 candidates applied to the first edition and 74 to the second. This difference may be justified by the better economic situation in Portugal by the end of 2018 than in the same period of 2017 (7% vs 8,9% unemployment).

The fact that Coimbra region is experiencing a period of increase of new IT companies demanding for professionals, and that there is a surplus of former students from other areas coming from the many higher education institutions in the region, results in a dual motivator for AeT candidacy: a surplus of people from areas with less employability, knowing that they will be easily placed in the work market as soon as they finish the programme.

As expected, the profiles of the candidates was very diverse, as can be seen from Table 1.

The original area of the candidates was very diverse ranging from affine engineerings to completely unrelated activities such as psychology or even social animation. This diversity suggests that initiatives such a AeT not only are relevant to a very broad public, but also that they are needed as an opportunity for people wanting to radically change careers.

Their academic degrees ranged from Master (or even PhD - 1 candidate included in the MSc column of Table 1) to Bachelor (*Licenciatura*, in Portugal) or even without any degree; their areas of study were both in Science, Technology, Engineering and Mathematics (STEM) and non-STEM (e.g. in Psychology, Management, Sports or Nursing); they were aged between 21 and 49.

As observe, the majority of the candidates already had an academic degree, although about 20% did not finish their Bachelor. Interesting is the fact that half come from non-STEM areas. Almost half of the candidates are aged between 30 and 40, and a few are more than 40 years old. As can be traditionally observed in any STEM-related programme, the majority of the candidates are male. It was also noted that a large percentage of the candidates (35% in the 1st edition and 25% in the 2nd) had a job and nevertheless were applying for this programme. The main reason was that they were not satisfied with their working conditions. For the purpose of candidate selection, we resorted to the help of a company specialising in Human Resources management, with a history of close collaboration with some of our industrial partners. This experience was leveraged for the benefit of the candidate selection process. After an initial period of publicity and marketing, candidate applications were received and the candidate selection process ensued, following these steps:

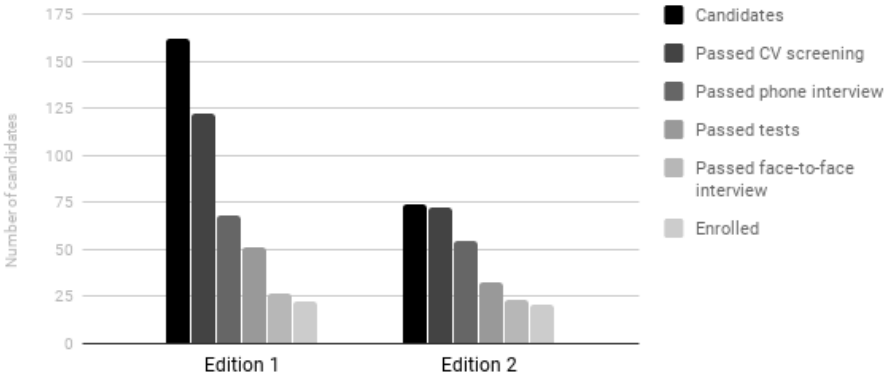


Figure 1 Candidates in each selection step.

Table 2 Profile of accepted candidates.

	Academic degree			Previous studies in STEM?		Gender		Age		
	no	Bac	MSc	Yes	No	M	F	< 30	< 40	≥ 40
Edition 1	15%	62%	23%	65%	35%	85%	15%	52%	44%	4%
Edition 2	22%	61%	17%	61%	39%	69%	31%	52%	29%	19%

1. *CV Screening:* An initial CV screening eliminated candidates that did not meet the requirements.
2. *Phone Interview:* A phone call interview allowed clarification of candidate profile and motivation, allowing further pruning of the candidate pool.
3. *Tests:* Selected candidates were invited for a session of psycho-technical tests and team exercises.
4. *Interview:* Finally those candidates that passed the tests went to a face-to-face interviews with a HR (Human Resources) specialist and a professor associated with the course.

In these tests and interviews, candidate capabilities, motivations and expectations were assessed. Candidates could also understand what was expected from them, if they were selected. Figure 1 shows the number of candidates that passed each step of the selection process. Starting from 162 and 74 candidates, respectively in the first and second edition, 26 and 23 were selected, from which 22 and 20, respectively, formally enrolled in the programme.

Despite the larger number of candidates in the 1st edition than in the 2nd, the number of accepted candidates was quite similar. It seems that the number of initial candidates makes little difference when selecting only those that are apt for the course. Taking a look at the profile of the candidates of both editions in Table 1, they are very similar. We then checked the profile of the accepted candidates (see Table 2), and compared with the initial candidates Table 1.

There seems to be no significant difference regarding their academic degree and gender, meaning that these factors do not influence the probability of a candidate being selected. However, candidates with previous studies in STEM seem to be in an advantage: about 25% of the candidates with STEM background from both editions were selected, as opposed to 16% of the candidates without such background. Also younger candidates look like having a higher probability of being selected.

3.2 Results from first edition

We received 22 highly motivated students, that were ready to work hard every day to achieve their goal of getting an internship in one of the partner companies. Some of them even dropped their stable jobs for a career change.

The instructors proved to be an invaluable piece to keep the pace of the course. They also responded to the exercises that the students submitted to the GitHub platform, providing a fast feedback. This worked reasonably well. However, there was one undesired side-effect: students started to try to solve the exercises for grading before time, losing the invaluable exercises for practice. To mitigate this, the exercises where organised into a progressive larger one, so that students would not be able to complete the last part (for grading) without going through the first part (for practice).

Student's grading was based on three elements:

- **Daily exercises**, specific to the subject addressed in the morning, for individual solving
- **Periodic projects**, specific to one subject, in teams of two, rotating for each new project
- **Final project**, executed in 3 weeks by teams of two, at the end of the academic phase of the programme.

The periodical projects were subjected to a presentation where students were required to explain and defend their solution. During the programme, many difficulties where faced and some important decisions where taken to overcome them. Some of the major difficulties and consequent decisions were:

- The pace of the programme was very intense, leading most of the students to manifest exhaustion. The pace had to be slowed down and pauses were introduced (like Easter break) to provide extra support to the students. Nevertheless, the planned syllabus was fully achieved.
- Some of the students showed serious difficulties in keeping up with the subjects. Using the daily exercises, it was possible to keep an up-to-date idea of the performance and difficulties. Individual recovery plans were defined for students showing less performance. This plan consisted of a set of materials and exercises specifically and individually tailored to each of these students.
- There was no break in lessons for the students to execute the periodic projects. The intention was to have the students to attend classes and do daily exercises in parallel with these projects, forcing them to manage their available time and keeping pressure. What frequently happened was that the students used the time they should spend in the daily exercises to solve the periodic project and disregarded the subjects being taught. To solve this, the periodic projects were rescheduled and every available break in classes (holidays, free afternoons, etc.) were used to minimise this issue.

At the end, (Figure 2), from the 22 enrolled students 18 passed (82% approval rate) which was quite good, considering that the majority of the students had not had any previous contact with programming. Nevertheless, better results were expected due to the nature of the programme and the motivation of the students.

From the first edition of AeT, the following conclusions where drawn:

- The introduction of new topics and its consolidation must be carefully balanced, so the students are able to assimilate them (through daily exercises) without burning out;
- Periodic projects help consolidating the subjects, but need time;
- A final project motivates students, and allows them to have better results. However they can work around difficulties, avoiding challenging subjects. Final project cannot, thus, replace the periodic projects.

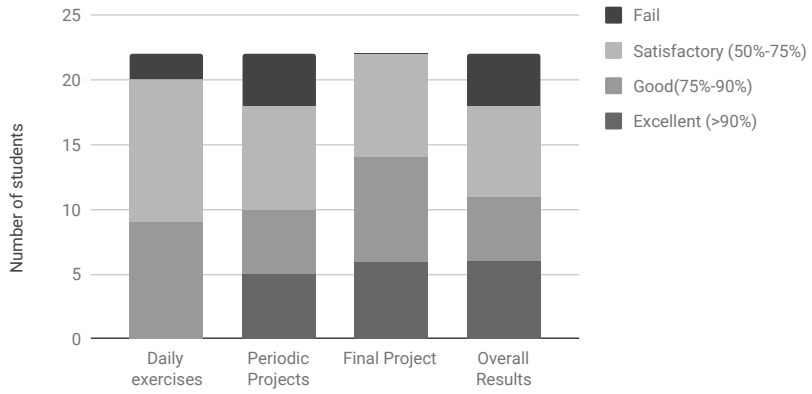


Figure 2 Students results – 1st Edition.

3.3 Results from second edition

Learning from the first edition experience, another approach was sought that would keep up the pace without over-stressing students and instructors with the daily evaluation. Three important changes were made:

1. We kept the daily exercises, but instead of using them for grading, they were only used for practice and feedback (students were required to submit them so the instructors could examine the code and provide feedback).
2. We introduced small projects that were announced tendentiously at the end of the week and to submit either at the end of the day (after classroom time), or in a later day, depending on the size and complexity of the projects.
3. The students were given exclusive time (usually one week) to execute the periodic projects.

First of all, it is necessary to stress that students are always different, and a direct comparison of the results from the two editions with such small figures cannot be conclusive. After a few weeks, it was noticed that the commitment from the 2nd edition students was quite lower than what was observed from the 1st edition. It was understood that since the students did not have the daily exercises for grading, they did not push their studies so hard, and started getting behind the imposed rhythm. Another observation was that they shared too much of their code, going beyond a healthy teamwork and discussion. Attending this, we decided for a fourth change, as a way to force them to study and to better assess their individual knowledge:

4. After each major topic, the students had to answer a written test (no computer).

This was quite the opposite of the initial idea of project-based learning. However, this was effective in motivating the students to continuously study and acquire the necessary competences.

Due to diverse reasons, four students abandoned the programme during the academic phase. One wasn't able to keep up with the pace. Another tried to simultaneously keep a part-time job in spite of being advised against it, and ultimately failed to cope. Two other students abandoned after prolonged absence due to health issues. The latter three could not have been anticipated in the recruitment process and can be considered fortuitous failures somewhat unrelated to the specifics of the course.

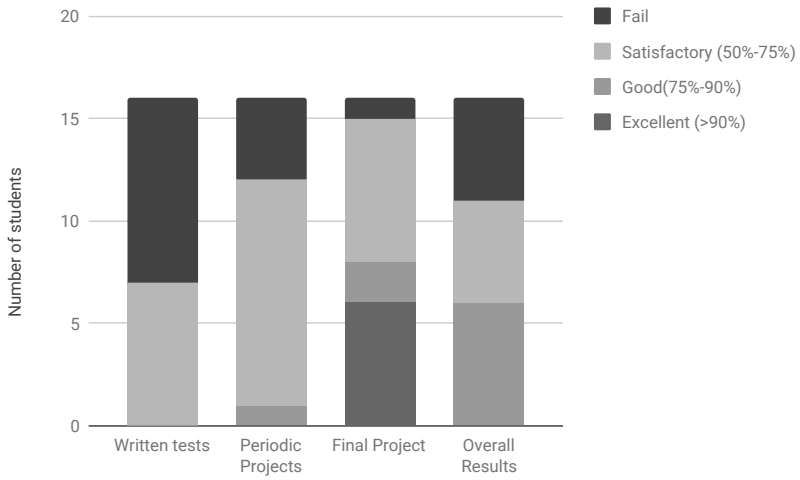


Figure 3 Students results – 2nd Edition.

Table 3 Students preferred internship location.

Location	Aveiro	Aveiro	Coimbra	Coimbra	Coimbra	Coimbra	Porto	Porto	Braga	Lisbon	Castelo Branco
Partner	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11
Edition 1	1	0	12	1	1	2	0	0	0	1	0
Edition 2	-	-	8	-	-	-	1	-	0	0	2

3.4 Placement and employees feedback

The 1st edition of AeT started with 11 industrial partners, that committed to receive at least one of the students in a professional internship context, offering a total of 35 internship proposals. The 2nd edition started with 6 industrial partners that proposed a total of 22 internships. The partners agreed to internship proposals that were in accordance to the programme contents.

Most of these industrial stakeholders operate in Coimbra but some internship proposals have been presented to Porto, Aveiro, Braga and Castelo Branco, outside the region of Coimbra. The industrial partners played a relevant role in trying to attract students to their internship proposals. At the end of the academic phase, the students had the opportunity to choose their preferred internship, with those with better grades being able to be the first to choose. The aim of this methodology is to make industrial partners pitch for the better classified students. Table 3 describes the students' placing by industrial partner as well as their location for both AeT editions.

The first conclusion from Table 3 is that there is a clear preference for Coimbra located internships. Within the Coimbra placements, the clear student preference is for the partners that offered the best perspectives of future integration and additional training and support. At the time of partners pitches, this was a clear students' concern. Students queries to industrial partners representatives where often related with further training opportunities. It's also relevant to point out that industrial partner #11 made a quite different approach to its pitch from Edition 1 to Edition 2 with clear results.

4 Conclusion

From this experiment, and observation of the results of both editions of the course, we learned the following lessons:

- Students need to be pressed – a strong pace, many exercises and frequent feedback pushes the students to work harder and learn faster.
- The high intensity nature of the course makes it necessary to continuously track and monitor student progress. This demands faculty to be highly available and supportive, making strong staff commitment crucial for success.
- Strategies must be found to maintain the strong pace consistently throughout the course, while ensuring that stress levels are under control and the workload manageable, both for students and staff.
- Grading is a strong incentive – the ultimate goal of the students is to get an internship in a good company, and eventually get a job there. The open box environment fosters both cooperation and competitiveness in day-to-day activities, while high priority in internship selection (because students choose internships in descending grade order) offers a longer-term incentive.
- A strong student motivator is the personal financial commitment to the course. This can be established by direct comparison with the motivation levels of students in courses of similar nature where tuition fees were sponsored by a third-party such as re-qualification grants.
- Although the selection process was in a large measure successful, it didn't take long after the course started to identify that a very small number of the accepted students would find it very difficult to meet the demands of the course, suggesting that the recruitment process could be improved.

The AeT course provides competences identified as priority by our industrial partners. A significant majority of students that attended the course have shown the ability to acquire these competences and were integrated as full-time collaborators. Although it requires significant engagement from both students and staff, AeT's primary objective, opening up new or better career opportunities for students, as well as addressing deficient supply of qualified professionals in the area, has been fully met.

References

1. J. Pedro Amaro, Jorge Barreiros, Fernanda Coutinho, João Durães, Frederico Santos, Ana Alves, Marco Silva, and João Cunha. Embedded programming bootcamp for career change. In *AmiES - International Symposium on Ambient Intelligence and Embedded Systems*, Coimbra, Portugal, 2019.
2. R. Barry. *FreeRTOS reference manual: API functions and configuration options*. Real Time Engineers Limited, 2009.
3. Sonal Dekhane, Kristine Nagel, and Nannette Napier. Summer programming boot camp: A strategy for retaining women in it. In *Proc. 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, page 678, NY, USA, 2015. Association for Computing Machinery.
4. B. Kernighan and D. Ritchie. *C Programming Language*. Prentice-Hall, USA, 1978.
5. Edward Ashford Lee and Sanjit Arunkumar Seshia. *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*. The MIT Press, 2nd edition, 2016.
6. I. Scott MacKenzie. *8051 Microcontroller*. Prentice Hall PTR, USA, 3rd edition, 1998.
7. Muhammad Ali Mazidi, Shujen Chen, and Eshragh Ghaemi. *STM32 Arm Programming for Embedded Systems (Volume 6)*. MicroDigitalEd.com, 2018.

8 Ian Sommerville. *Software Engineering*. Addison-Wesley Publishing, USA, 2010.

9 Bjarne Stroustrup. *C++ Programming Language*. Addison-Wesley Professional, 2013.

10 Kyle Thayer and Andrew J. Ko. Barriers faced by coding bootcamp students. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*, ICER '17, page 245–253, NY, USA, 2017. Association for Computing Machinery.

11 Yu-Cheng Tu, Gillian Dobbie, Ian Warren, Andrew Meads, and Cameron Grout. An experience report on a boot-camp style programming course. In *49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, NY, USA, 2018. Association for Computing Machinery.

An Experience of Game-Based Learning in Web Applications Development Courses

Míriam Antón-Rodríguez
University of Valladolid, Spain
mirant@tel.uva.es

María Ángeles Pérez-Juárez
University of Valladolid, Spain
mperez@tel.uva.es

Francisco Javier Díaz-Pernas
University of Valladolid, Spain
pacper@tel.uva.es

David González-Ortega
University of Valladolid, Spain
davgon@tel.uva.es

Mario Martínez-Zarzuela
University of Valladolid, Spain
marmar@tel.uva.es

Javier Manuel Aguiar-Pérez
University of Valladolid, Spain
javagu@tel.uva.es

Abstract

Preparing graduates for working in the software engineering industry is challenging and requires effective learning frameworks and methodologies. More specifically, the challenge of teaching programming languages and paradigms is a very complex task that needs innovative educational tools. This paper presents a game-based educational tool named eLiza, developed and used to support the teaching and learning of programming languages and paradigms related to the development of web applications. eLiza was initially developed as a Moodle-based web application because Moodle is the educational eLearning platform used at the University of Valladolid, but as the use of mobile devices is constantly increasing, Android and iOS versions were created later in order to facilitate the access of the students to the games. This paper describes the main elements and the mechanics in playing eLiza. And it also describes an experience of its use in two engineering courses related to web programming applications development, offered to students in two different engineering study programs at the University of Valladolid, during the academic years 2017-2018 and 2018-2019. The great majority of the students, more than 75%, considered that the use of the eLiza game-based educational tool was positive to improve the teaching and learning process of the topics covered by the courses.

2012 ACM Subject Classification Social and professional topics → Computer science education

Keywords and phrases eLearning, mLearning, Game-based Learning, Programming Languages, Web Applications Development

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.3

1 Introduction

There are a lot of courses whose objective is to teach different programming languages and paradigms to the students. These courses are mainly offered to University students. The learning of programming languages and paradigms is tough and requires a lot of practice.

First International Computer Programming Education Conference (ICPEC 2020).
Editors: Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões; Article No. 3; pp. 3:1–3:11

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

This discipline is also very different to others in which students take memorization procedures as a base. Programming requires a lot of additional work which is developed within the classroom, especially when compared to more theoretical fields of study [4]. According to Vega et al. [19] students have the perception that programming is difficult and it is quite frequent to hear about problems related to frustration and lack of motivation. Moreover, programming is hard to teach, in fact, there are professors that think that programming requires abilities more than knowledge [17].

In such a complex context, gamification can play an important and positive role. Teachers can use game mechanics in non-game contexts to engage students in solving problems and increase their motivation and academic performance. Students who participate in games develop more intellectual abilities than those who do not [6]. Some studies have shown that the part of spare time that students spend gaming exceeds the part of spare time that they spend watching television [7]. For this reason, to take advantage of the benefits of games for educational purposes would open a lot of new possibilities.

In order to explore the possibilities of using game-based educational tools in the formal teaching of programming languages and paradigms, a game-based educational tool eLiza has been developed and tested in different courses offered at the University of Valladolid for engineering students. eLiza was initially developed as a Moodle-based tool. The reason is that Moodle is the educational eLearning platform used by the University of Valladolid to support the teaching and learning process in formal courses. Moreover, more recently Android and iOS versions have also been implemented to facilitate the access of the students to the games.

eLiza uses different strategies to increase students' motivation. The objective is that students really enjoy themselves as they do when they play a game they like in their spare time. The experiences have been carried out in two engineering courses, related to web programming applications development, during the academic years 2017-2018 and 2018-2019. The analysis of the experiences has been based in both qualitative and quantitative procedures.

Therefore, the objective of this paper is to present a game-based educational tool that aims to support the teaching and learning process of programming learning and paradigms. As well as to test the tool in a university academic environment in order to determine the usefulness of gamification and more specifically of this educational tool in such a context for both students and teachers.

2 Teaching Programming Languages and Paradigms

Besides the inherent difficulty of programming as a discipline, the focus of the problem could be in the use of inefficient and inadequate methodologies and educational tools to teach this subject. For this reason, during the last decades, researchers have searched for ways to improve the academic performance of students, especially in the case of newcomer students [12].

With such a complex context it is important to have a clear view of which are the main problems that the students face when approaching to the theory and practice of the programming languages and paradigms. The educational tools oriented to learning programming should take into account theses needs and provide resources and strategies to manage them.

The first main issue is the fact itself of having to use technology-based educational tools. At this moment, the focus is not on the use of the computer itself as younger generations are used to use them, but on the anxiety that generates the use of certain complex software

applications. Anxiety and a negative attitude negatively affect learning and conditions the use of the computer [13]. For this reason, it is important that students can quickly and easily adapt to these type of systems.

A first conclusion is that a good educational tool focused on programming must be easy to use so the students invest their effort and time in learning programming languages and paradigms and not in learning how to use the tool itself. Another central issue is motivation. Students must spend lots of hours practising if they want to learn how to program, and this is not possible if they are not very motivated [11]. Students and teachers both think that programming is difficult to learn, specially for new students. Students must be adequately motivated during the whole process. Improving the practises of teaching and learning programming is not possible without paying special attention to the motivation of students for learning [8]. According to Brito & Sá-Soares [4], motivation is the most important factor to succeed when learning how to program. A motivated student will succeed no matter other circumstances including a bad teacher or a bad structure of the course. In the same way, if a student is not motivated he will probably not succeed no matter how favourable are other circumstances.

Most teachers apply different approaches to support the learning process of students and to adequately motivate them. Most times this approach is based in the use of technology [18]. The results of different studies also suggest that technology-based educational tools that are easy to use can improve motivation and efficiency in the learning process [11].

When teaching programming languages and paradigms it is very important to identify concepts with very precise definitions. Students must understand concepts in order to be able to solve programming challenges [9].

In order to prepare students for programming it is important that they first know the basic concepts of algorithmic thinking. This means that students must be able to clearly define a problem, to divide it into smaller parts that are easier to solve and to determine the steps to solve the entire problem. In order to fulfil this objective students must be able to distinguish the essential characteristics from the unnecessary details [5].

So programming requires that students understand the problem, define the solution and finally translate the solution into code by using a programming language. Different studies show that students find serious difficulties in every step of this process [10].

During the last years, teachers and researches had tried to improve the teaching of programming. To achieve this objective, they have focused on the different elements of the process. One of the aspects in which they have focused is methodologies. In fact, choosing the right methodology to teach programming is one of the main issues of the debate of teaching programming languages and paradigms [1, 2, 3].

A challenge is also to convert programming into an activity that is mainly developed in groups instead of individually. With the idea of promoting workgroup in the learning process of languages and paradigms some projects have been developed like Nucleo [15]. This project promotes that students acquire social skills and abilities for working in groups and that they adopt a more active role in the learning process. This is important as in the software industry projects are mainly undertaken by groups and not by individual programmers.

3 eLiza

eLiza is a game-based educational tool initially developed for the educational eLearning platform Moodle. eLiza is a competitive game which main objective is that students learn while playing. To achieve this goal, eLiza challenges students with questions classified in different levels of difficulty. eLiza promotes competitiveness among students by offering information about the students' performance.

eLiza is expected to have a positive impact into the learning process by increasing the involvement of students. Teachers can establish prizes at different score levels, so students are encouraged to reach those score levels. Moreover, teachers can use the results obtained by students in eLiza as an assessment element to evaluate them. On the other side, students can use the educational game for their own self-assessment. eLiza was initially developed as a module for the educational eLearning platform Moodle. For this reason, eLiza has been mainly used from desktop browsers, but as it does not have any special requirements, it could have been played from browsers in mobile devices such as tablets and smartphones. Moreover, as the use of mobile applications has grown exponentially in recent years [14], Android and iOS versions have recently been created in order to facilitate the access of the students to the games.

When accessing eLiza, the teacher will see a main screen with five buttons which give him access to the different sections of the application: Let's Play, Let's Play in Groups, Management, Add a new Question and Statistics. The student can access the same sections that the teacher, except for the Management section (see Figure 1).

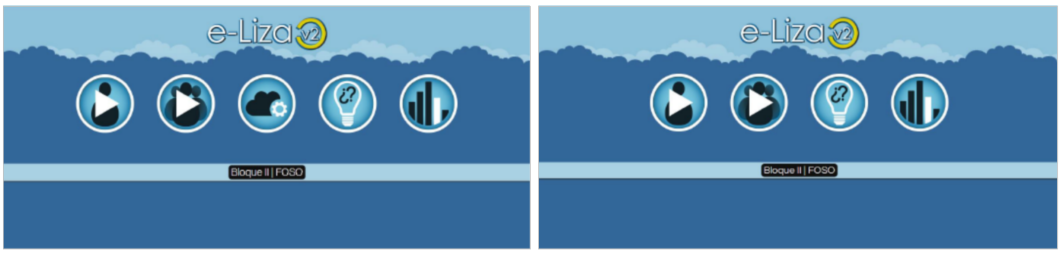


Figure 1 eLiza main access screen for teachers (left) and students (right).

The teacher can manage the questions bank of a game. As shown in Figure 2, to add a new question it is only necessary to complete a form providing the name, the type of question (multiple choice or true/false), the content and the possible answers indicating for each case if the answer is a correct or an incorrect one. Another element that is necessary to define a question is the labels associated to that question. It is possible to add as many labels as desired in order to have the question adequately categorized. Labels must have previously been created. Finally it is necessary to indicate the time the student will have to answer the question during the game, and the value of that question in points, that is the points that the student will be assigned during the game in case he answers the question correctly. Regarding the value of the question it is important to mention that it is also possible to assign a penalty which is a number that indicates the percentage of the value of the question in points that will be subtracted in case the student answers that question incorrectly during the game.

The teacher can also view and manage all the questions proposed by the students for a course. Each of the questions proposed by the students has a state that can be: pending (the question is still waiting for the teacher's approval), approved (the question has been accepted by the teacher) and rejected (the question has been rejected by the teacher). The teacher can see the content of any of the pending questions in order to decide to approve or to reject the question.

Figure 2 Questionnaire to add a new question.

In the Statistics section, the teacher can see the activity of the students in the games. The global statistics include all the students that have participated in the games of a course. The systems presents these statistics by using two type of graphs that are the pie chart and the temporary graph (see Figure 3). The pie chart shows more clearly the percentage of success, failures and not answered questions. While the temporary graph shows more clearly the average score of the students of the course, grouped by months during the period in which those students have taken part in games. The teacher can also view the statistics per student, the statistics per game and the statistics per question.

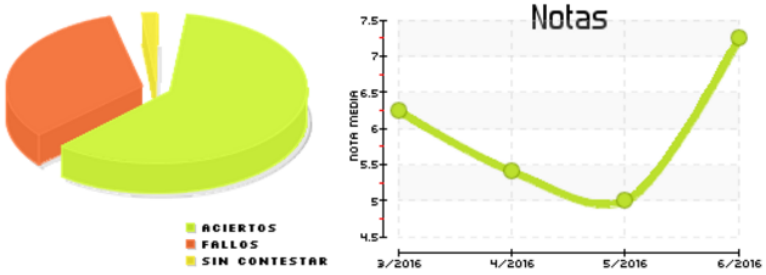


Figure 3 Global statistics (all students & all games).

The student has also access to different functionalities which are summarised in the eLiza main screen shown to him when he accesses the application. One of the most important buttons for the student is the Let's Play button which allows him to start a new game. When the student starts the game he must answer all the questions. After finishing the game the student can see the result including the number of questions answered correctly, the number of points obtained over the maximum number of points that is possible to obtain, and his final score which is the number of points obtained weighted over ten (see Figure 4).

Another possibility available for the student is to play in group. First the student will see the list of all the group games in which he can take part, for example in the case he has been challenged by other student. As in an individual game the student has to answer the different questions proposed within the time provided for each question. Any student can

Figure 4 Information offered to a student after finishing a game.

create a group game indicating the students that are challenged selected from the list of students enrolled in the course. The system announces the winner of a group game only after all the students challenged in that group game has played.

Finally the student has the possibility to access the statistics about his own participation in the different games (see Figure 5. In this case the student can see information about the percentage of times he answered questions correctly and incorrectly, the percentage of times he did not answer a question, the number of complete and incomplete games, that is the number of games in which he answered all the questions, and the number of games in which he did not answered all the questions foreseen. Finally the student can also view the average time that took him to answer a question, and the average and maximum scores that he obtained in the games he took.

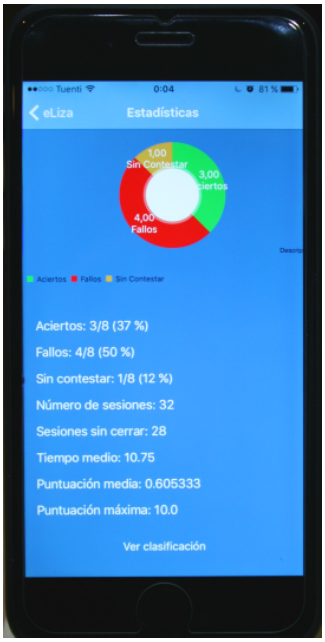


Figure 5 Student's statistics in the eLiza iOS version.

As explained before, eLiza is also available as Android and iOS native applications for mobile devices. The reason to develop also these versions, apart from the desktop version, was to facilitate the access of the students to the games. Most of the students have their own laptops and carry them to the University. However, for obvious reasons, laptops are not continuously turned on as it happens with mobile devices.

4 Results

The courses in which the game-based educational tool eLiza has been used are courses focused on programming languages and paradigms that are part of different engineering study programs offered by the University of Valladolid. The experiences have been carried out during the academic years 2017-2018 and 2018-2019. The students were encouraged to use eLiza and then were invited to talk about their user experience, the benefits obtained, the problems found, the upgrades suggested, etc., in a forum. The great majority of the students, more than 75%, considered the use of the eLiza game-based educational tool was positive to improve the teaching and learning process of the topics covered by the course. Moreover, twenty-five students that participated in the experience using the Android version of eLiza, completed the MARS (Mobile Application Rating Scale) questionnaire [16]. The global results of the experience are shown in Table 1.

Table 1 Global statistics of the experience with the Android version of eLiza.

Statistics Item	Value
Successes	1891/2504 (75.52%)
Faults	588/2504 (23.48%)
Unanswered	25/2504 (1%)
Number of sessions	513
Number of sessions per user	20.52
Number of sessions without closing	22
Average response time	19.64 s.
Average grade	7.1
Maximum grade	10

The results of the survey for each question were the following:

1. Entertainment: Is the app fun/entertaining to use? Does it use any strategies to increase engagement through entertainment (e.g. through gamification)?

   - Dull, not fun or entertaining at all ..... 0.0 %
   - Mostly boring ..... 12.5 %
   - OK, fun enough to entertain user for a brief time (< 5 minutes) ..... 37.5 %
   - Moderately fun and entertaining, would entertain user for some time (5-10 minutes total) ..... 50.0 %
   - Highly entertaining and fun, would stimulate repeat use ..... 0.0 %

2. Interest: Is the app interesting to use? Does it use any strategies to increase engagement by presenting its content in an interesting way?

   - Not interesting at all ..... 0.0 %
   - Mostly uninteresting ..... 12.5 %
   - OK, neither interesting nor uninteresting; would engage user for a brief time (< 5 minutes) ..... 25.0 %
   - Moderately interesting; would engage user for some time (5-10 minutes total) ..... 62.5 %
   - Very interesting, would engage user in repeat use ..... 0.0 %

3. Interactivity: Does it allow user input, provide feedback, contain prompts (reminders, sharing options, notifications, etc.)? Note: these functions need to be customisable and not overwhelming in order to be perfect.

   - No interactive features and/or no response to user interaction ..... 0.0 %
   - Insufficient interactivity, or feedback, or user input options, limiting functions 0.0 %
   - Basic interactive features to function adequately ..... 25.0 %
   - Offers a variety of interactive features/feedback/user input options ..... 37.5 %
   - Very high level of responsiveness through interactive features/feedback/user input options ..... 37.5 %

4. Target group: Is the app content (visual information, language, design) appropriate for your target audience?

   - Completely inappropriate/unclear/confusing ..... 0 %
   - Mostly inappropriate/unclear/confusing ..... 0 %
   - Acceptable but not targeted. May be inappropriate/unclear/confusing ..... 0 %
   - Well-targeted, with negligible issues ..... 50 %
   - Perfectly targeted, no issues found ..... 50 %

5. Performance: How accurately/fast do the app features (functions) and components (buttons/menus) work?

   - App is broken; no/insufficient/inaccurate response (e.g. crashes/bugs/broken features, etc.) ..... 0 %
   - Some functions work, but lagging or contains major technical problems ..... 0 %
   - App works overall. Some technical problems need fixing/Slow at times ..... 0 %
   - Mostly functional with minor/negligible problems ..... 50 %
   - Perfect/timely response; no technical bugs found/contains a “loading time left” indicator ..... 50 %

6. Ease of use: How easy is it to learn how to use the app; how clear are the menu labels/icons and instructions?

   - No/limited instructions; menu labels/icons are confusing; complicated ..... 0.0 %
   - Usable after a lot of time/effort ..... 0.0 %
   - Usable after some time/effort ..... 12.5 %
   - Easy to learn how to use the app (or has clear instructions) ..... 12.5 %
   - Able to use app immediately; intuitive; simple ..... 75.0 %

7. Navigation: Is moving between screens logical, accurate, appropriate, uninterrupted; are all necessary screen links present?

   - Different sections within the app seem logically disconnected and random/confusing/navigation is difficult ..... 0 %
   - Usable after a lot of time/effort ..... 0 %
   - Usable after some time/effort ..... 0 %
   - Easy to use or missing a negligible link ..... 50 %
   - Perfectly logical, easy, clear and intuitive screen flow throughout, or offers shortcuts ..... 50 %

8. Layout: Is arrangement and size of buttons/icons/menus/content on the screen appropriate or zoomable if needed?

   - Very bad design, cluttered, some options are impossible to select/locate/see/read device display not optimised ..... 0.0 %
   - Bad design, random, unclear, some options difficult to select/locate/see/read . 0.0 %
   - Satisfactory, few problems with selecting/locating/seeing/reading items or with minor screen-size problems ..... 12.5 %
   - Mostly clear, able to select/locate/see/read items 37.5%
   - Professional, simple, clear, orderly, logically organised, device display optimized. Every design component has a purpose. 50.0%

9. Graphics: How high is the quality/resolution of graphics used for buttons, icons, menus, content?

   - Graphics appear amateur, very poor visual design; disproportionate, completely stylistically inconsistent ..... 0.0 %
   - Low quality/low resolution graphics; low quality visual design; disproportionate, stylistically inconsistent ..... 0.0 %
   - Moderate quality graphics and visual design (generally consistent in style) ... 37.5 %
   - High quality/resolution graphics and visual design; mostly proportionate, stylistically consistent ..... 50.0 %
   - Very high quality/resolution graphics and visual design; proportionate, stylistically consistent throughout ..... 12.5 %

10. Visual appeal: How good does the app look?

    - No visual appeal, unpleasant to look at, poorly designed, clashing/mismatched colours ..... 0 %
    - Little visual appeal; poorly designed, bad use of colour, visually boring ..... 0 %
    - Some visual appeal; average, neither pleasant, nor unpleasant ..... 50 %
    - High level of visual appeal – seamless graphics – consistent and professionally designed ..... 25 %
    - As above + very attractive, memorable, stands out; use of colour enhances app features/menus ..... 25 %

From the results of the study a series of conclusions were drawn. The majority of the students considered the app was highly entertaining and fun, but a small percentage considered that the app was boring or only entertaining for a brief time. None of the students considered that the app was very interesting, but most of them considered it to be moderately interesting. Most students considered the app to be interactive, but a small group commented that the interactivity characteristics available at the app were very basic. Most students thought that the app was well targeted and considered that the functions available worked correctly, or else encountered minor problems when using it. Finally, most students considered the application to be very easy to use and that the navigation was perfectly logical, or at least easy enough to understand and navigate.

References

1 Owen Astrachan, Kim Bruce, Elliot Koffman, Michael Kölling, and Stuart Reges. Resolved: Objects early has failed. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '05, page 451–452, New York, NY, USA, 2005. Association for Computing Machinery. doi:10.1145/1047344.1047359.

2 Frances Bailie, Mary Courtney, Keitha Murray, Robert Schiaffino, and Sylvester Tuohy. Objects first - does it work? *Journal of Computing Sciences in Colleges*, 19(2):303–305, December 2003.

3 John Bergin, Jutta Eckstein, Mary Lynn Manns, and Eugene Wallingford. Patterns for gaining different perspectives. In *8th conference on Pattern Languages of Programs (PLoP 2001)*, Illinois, USA, 2001.

4 Miguel A. Brito and Filipe De Sá-Soares. Assessment frequency in introductory computer programming disciplines. *Computers in Human Behavior*, 30:623–628, January 2014. doi:10.1016/j.chb.2013.07.044.

5 Jill Denner, Linda Werner, and Eloy Ortiz. Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers and Education*, 58:240–249, 2012.

6 Juan Alberto Estallo. *Los videojuegos. Juicios y prejuicios.* Planeta, Barcelona, 1994.

7 Francisco José García-Peñalvo, Mark Johnson, Gustavo Ribeiro Alves, Miroslav Minović, and Miguel Ángel Conde-González. Informal learning recognition through a cloud ecosystem. *Future Generation Computer Systems*, 32(C):282–294, March 2014.

8 Tony Jenkins. The motivation of students of programming. In *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '01, page 53–56, New York, NY, USA, 2001. Association for Computing Machinery. doi:10.1145/377435.377472.

9 Cagin Kazimoglu, Mary Kiernan, Liz Bacon, and Lachlan MacKinnon. Learning programming at the computational thinking level via digital game-play. *Procedia Computer Science*, 9:522–531, 2012. doi:10.1016/j.procs.2012.04.056.

10 Maria Kordaki. A drawing and multi-representational computer environment for beginners' learning of programming using c: Design and pilot formative evaluation. *Computers & Education*, 54(1):69–87, 2010. doi:10.1016/j.compedu.2009.07.012.

11 Kris M.Y. Law, Victor C.S. Lee, and Y.T. Yu. Learning motivation in e-learning facilitated computer programming courses. *Computers & Education*, 55(1):218–228, 2010. doi:10.1016/j.compedu.2010.01.007.

12 Jan Moons and Carlos [De Backer]. The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism. *Computers & Education*, 60(1):368–384, 2013. doi:10.1016/j.compedu.2012.08.009.

13 Aysen Gurcan Namlu. The effect of learning strategy on computer anxiety. *Computers in Human Behavior*, 19(5):565–578, 2003. doi:10.1016/S0747-5632(03)00003-7.

14 William T Riley, Daniel E Rivera, Audie A Atienza, Wendy Nilsen, Susannah M Allison, and Robin Mermelstein. Health behavior models in the age of mobile interventions: are our theories up to the task? *Translational Behavioral Medicine*, 1(1):53–71, February 2011. doi:10.1007/s13142-011-0021-7.

15 Pilar Sancho-Thomas, Rubén Fuentes-Fernández, and Baltasar Fernández-Manjón. Learning teamwork skills in university programming courses. *Computers & Education*, 53(2):517–531, 2009. doi:10.1016/j.compedu.2009.03.010.

16 Stoyan R Stoyanov, Leanne Hides, David J Kavanagh, Oksana Zelenko, Dian Tjondronegoro, and Madhavan Mani. Mobile app rating scale: A new tool for assessing the quality of health mobile apps. *JMIR mHealth uHealth*, 3(1):e27, March 2015. doi:10.2196/mhealth.3422.

17 Jun Tan, Xianping Guo, Weishi Zheng, and Ming zhong. Case-based teaching using the laboratory animal system for learning C/C++ programming. *Computers & Education*, 77:39–49, 2014. doi:10.1016/j.compedu.2014.04.003.

18 Georgi Tuparov, Daniela Tuparova, and Anna Tsarnakova. Using interactive simulation-based learning objects in introductory course of programming. *Procedia - Social and Behavioral Sciences*, 46:2276–2280, 2012. doi:10.1016/j.sbspro.2012.05.469.

19 Carlos Vega, Camilo Jiménez, and Jorge Villalobos. A scalable and incremental project-based learning approach for cs1/cs2 courses. *Education and Information Technologies*, 18(2):309–329, June 2013. doi:10.1007/s10639-012-9242-8.

Use of Automatic Code Assessment Tools in the Programming Teaching Process

Marílio Cardoso[1]
Department of Informatics, ISEP, Polytechnic of Porto, Portugal
joc@isep.ipp.pt

António Vieira de Castro
Department of Informatics, ISEP, Polytechnic of Porto, Portugal
avc@isep.ipp.pt

Álvaro Rocha
Department of Informatics, Faculty of Science and Technology, University of Coimbra, Portugal
amrrocha@gmail.com

Emanuel Silva
Department of Informatics, ISEP, Polytechnic of Porto, Portugal
ecs@isep.ipp.pt

Jorge Mendonça
Department of Mathematics, ISEP, Polytechnic of Porto, Portugal
jpm@isep.ipp.pt

Abstract

The teaching of programming process is essential to prepare students for the development of computer applications and software solutions. During the last decade, a variety of tools facilitating automatic validation of programming code have been developed. In this context, authors start to analyze and studying some tools with this potential and a possible use with pedagogical purposes. For the last three years a study has been carried out related with the implementation of VPL (Virtual Programming Lab) a plug-in developed specifically for Moodle (Modular Object-Oriented Dynamic Learning Environment) on a Java-based programming discipline during the Informatics Engineering degree of the Informatics Engineering Department (DEI) from the School of Engineering of Polytechnic Institute of Porto (ISEP/P.PORTO). This paper will present how VPL was introduced and some results of this experiment before the implementation in the learning process of another tool (Mooshak) as a real-time automatic code evaluation. These tools allow to edit and execute programs, in a large range of languages, and enables automatic assessment and prompt feedback.

2012 ACM Subject Classification Software and its engineering; Applied computing → Interactive learning environments

Keywords and phrases Teaching programming, APROG, Moodle, VPL, Mooshak, Automatic assessment

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.4

1 Introduction and context

Programming is, nowadays an important skill even for those who do not work or intend not to work in information and communication technologies area (ICT).

Besides it could be relevant to personal development, it can also help in professional activity, allowing to better understand the context of technology, automate and optimize tasks.

[1] Corresponding author

First International Computer Programming Education Conference (ICPEC 2020).
Editors: Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões; Article No. 4; pp. 4:1–4:10

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1.1 Programming skills in modern society

Learning programming in early ages increases logical reasoning, improves cognitive abilities and human interactivity and transforms perception to establish logical connections giving also self-confidence and problem-solving skills [1].

On the other side, computer programming teaching is a complex and challenging task for any teacher, mainly for those who have beginner students, since this activity is very different from teaching any other subject, so the approach will be different.

1.2 The traditional programming Learning process

On the "write code" process, the programmer must "translate" the algorithm into a programming language. A programming language is a formal language with a specific set of well-defined instructions and syntax rules, which enable software development. There is a huge set of programming languages, each one with specific syntax and application fields.

According Cardoso et. al. [3], the TIOBE[2] index, that is a Programming Community index, is a good indicator of the popularity of programming languages and considering that it is one of the most cited related indexes, their information provides a global overview about the programming languages usage. Their ratings are monthly updated and are based on the number of skilled engineers world-wide, courses and third-party vendors.

Cardoso et al. also refer PYPL (PopularitY of Programming Language)[3], another index about programming languages that is based on Google and the Redmonk[4] that is a ranking of programming languages obtained from information about GitHub code lines and "StackOverflow"[5] language tags.

Nowadays, Java, Python and C# appear in the top 5 of recent rankings of these indices.

In Informatics Engineering degree of the Informatics Engineering Department of School of Engineering (ISEP), Polytechnic Institute of Porto (P. PORTO), Java was the chosen language to introduce novice students to programming.

APROG (Algorithms and Programming) is the first course unit that directly deals with programming skills, where students start coding with Java using NetBeans, an Integrated Development Environment (IDE).

1.3 The case of the APROG course unit

Figure 1 presents APROG organization with three different types of classes: theoretical (T), theoretical-practical (TP) and practical-laboratorial (PL).

This course unit takes place during the first semester of the first year and has usually about three hundred students and several teachers (usually between 6 and 10).

We must grant that lessons are similar inside each class and with its important to ensure that pedagogical goals will be the same, independently of the teachers allocated to each class. For this, a good planning and organization is needed as well as strong and clear coordination.

[2] https://www.tiobe.com/tiobe-index
[3] http://pypl.github.io/PYPL.html
[4] https://redmonk.com/
[5] https://pt.stackoverflow.com/

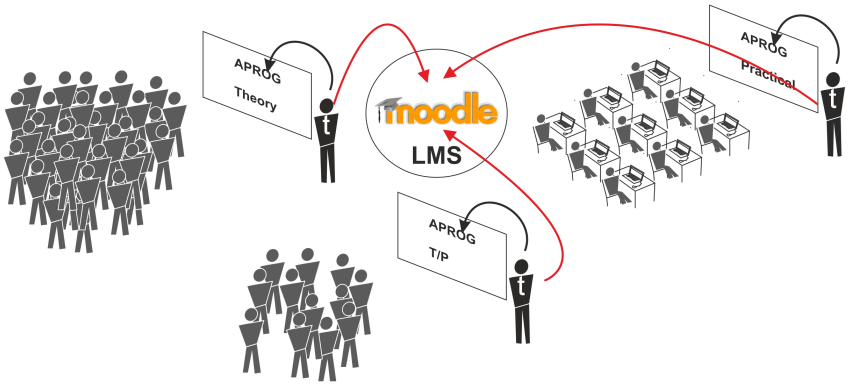


Figure 1 Types of APROG classes.

On the Table 1 are presented the different weekly type of classes as well as the number of hours allocated to it.

Table 1 Weekly class types.

Class type	Number of classes by week	Hours by class	Total week hours
T	1	1	1
TP	1	1	1
PL	2	2	4

In the last three academic years (2017–2018, 2018–2019 and 2019–2020), in APROG, were allocated respectively 10, 6 and 8 teachers for a population of 318 students in 2017–2018, 307 in 2018–2019 and 304 in 2019–2020.

The teacher responsible for APROG has changed on 2019–2020 and his responsibility is to teach theoretical classes. Other 3 teachers were involved in theoretical-practical classes and they also teaches practical-laboratorial classes. The distribution of teachers by type of class is shown in Table 2.

Table 2 Teachers by class type.

Class type	T	TP	PL
2017/18	1	3	9
2018/19	1	3	5
2019/20	1	3	7

In the practical-laboratorial classes a methodology named eduScrum [4] was applied where students were organized in work teams. EduScrum methodology is a variation of Scrum [12] but applied with pedagogical purposes.

Scrum is a method that has been used to manage the development of software and many other complex products, mainly by software companies to promote teamwork and increase productivity and creativity on the team members. Its use is being tested in many other areas, and education is one of them [6].

In Portugal, as well as in the rest of Europe, one of the most popular and widespread Learning Management System (LMS) is Moodle, that is an open source solution delivered under General Public Licenses (GNU). Since 2006 ISEP also adopted Moodle and it is widely

used to place organized educational contents. Moodle allows the submission of works that has been assigned to students and promote communication between teachers and students and do students surveys.

In APROG course unit, new content is weekly organized and in each week are included in Moodle a set of theoretical content and a set of practical exercises that students must solve.

In each class teacher analyzes the resolutions provided by the students and give them a feedback. Each class has about twenty students and each worksheet have six or more exercises. This activity represents a big challenge for APROG teachers, because the feedback must be provided weekly, resulting in a great overhead of tasks. With such a big set of tasks its almost impossible to evaluate deeply all of them. The existing solution is to evaluate by sampling, and this means that the feedback is incomplete and with some delay. This prevent students to continue developing their works and improve faster their programming skills.

The lack of feedback and mentoring from a teacher to students is one of the most important questions related with learning about programming [11], that is related and may contribute to students' lack of motivation and commitment.

We identified this as a problem and we wanted to contribute to solve it urgently since teachers do not have enough time to keep up with students, and also, students can practice coding without having to wait a few hours or even days for the teacher validation of their code.

Our first impulse was to propose a reduction of the number of students per class as well as the number of exercises that they must solve but, due the importance of practice for develop programming skills, teachers agreed to not reduce it.

So, we conclude that something should be done to help, and we began to study and identify new ways to reduce the time spent by teachers in verifying and evaluating assignments and helping students to improve their programming skills more quickly.

In this demand we identified a myriad of potential tools for our purpose. However, we defined a set of requirements, according to the conditions we had, the needs and characteristics that we thought desirable for the tool to choose. These requirements were: to be a free tool, integrated to Moodle, easy to use, suitable for teaching programming, allowing code upload, with plagiarism detection capabilities, with security concerns and supporting Java. With these characteristics we choose Virtual Programming Lab (VPL) to implement a study and realize a preliminary pedagogical experience.

2 The VPL case in APROG

As mentioned, to analyze the use of automatic code validation tools and the study of their potential, a pedagogical perspective started in a study that was based on the use of VPL.

2.1 The VPL plugin

VPL is a Moodle plugin developed by the Department of Computer Science and Systems from University of Las Palmas of Gran Canaria, Spain and it is an open source solution under the GNU/GPL license [11].

This is a tool that allows to manage programming assignments on a large set of programming languages. VPL can identify the language in an automatic way.

The VPL architecture has the following three main specifications: [2]

- a Moodle module with specific features such as: submission management, assessment support and anti-plagiarism;
- a browser-based code editor, which allows coding and execute programs;
- a Linux server (jail server) which hosts the environment where the students' assignment will be executed and evaluated in a remote and secure way.

With VPL its possible to manage computing programming assignments and provide a mechanism for automatic grading. Other functionalities allow to edit, compile, run, debug and evaluate code, with prompt feedback and at same time stores historic results related with compilation and implementation of the assignment and track the student's submissions.

2.2 Process Implementation

Several steps were panned to implement the process, as well a list of conditions that must be considered. One of the most important was that the experience must not disrupt the existing pedagogical process, that is, it should not disturb the normal course of classes and at same time, should assure impartiality on the evaluation to all students.

It was defined that VPL should be available in the institutional LMS (Moodle), so it installation was carried out by the responsible technician for the internal Moodle management.

All the steps were carefully planned to grant a soft, robust and progressive implementation of this study. On Figure 2 we present the main phases on the process.

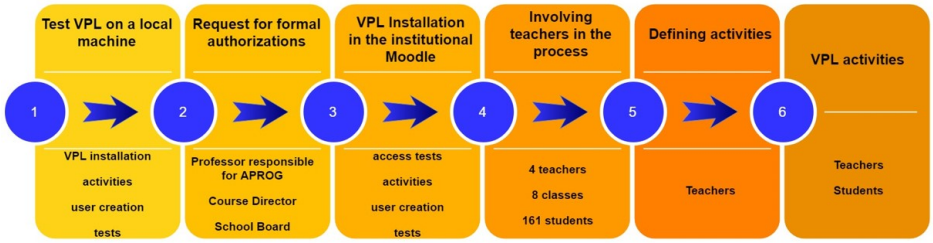


Figure 2 Planning process phases.

From the set of exercises on each week are defined some activities and created adequate assignments. After identifying the exercises, it was necessary to prepare their statements and test cases, implement and test with a dummy user with student role.

Finally, VPL will be available for all students and assignments will be possible.

2.3 VPL activities

Were selected six exercises to use VPL, and the necessary adaptations were made to the statement, in order to reduce, as much as possible, any ambiguities and referring some particularities to be considered in the submission. Examples of the input data format and the expected results have also been added, in order to smooth the use and reduce the probability of error.

The students prepared the resolutions of the exercises in the IDE and, for the mentioned six exercises, proceeded to their submission in the VPL. In each submission, the system gives feedback, allowing the student to see if the result obtained was the expected one and, if not, to identify the tests in which he/she failed, allowing him/her to redo the resolution and resubmit.

2.4 Some results related with VPL

VPL was used in the academic years 2017–2018, 2018–2019 and 2019–2020.

However, the presented results refer only to the academic year 2018–2019. This is due to the fact that, in that year, the number of students involved was significantly higher than in the previous year, the teachers had more experience using VPL and the (opinion)

survey carried out was answered by a larger number of students becoming more complete and consistent. Data of the academic year 2017-2018 were processed by Excel and SPSS (version 25).

With this study, the matrices of Spearman's nonparametric correlations and Cronbach's alpha were determined, this parameter being an indicator of the internal consistency of the survey. On the survey, to be considered reliable, Cronbach's alpha must be greater than 0.7 [10]. In the 2017–2018 survey, a Cronbach's alpha of 0.614 was obtained, thus showing a poor reliability, so we do not report these results. In the year 2018–2019 the survey was revised and adjusted so that a Cronbach's alpha of 0.822 was obtained, which reveals the reliability of the study and the results presented.

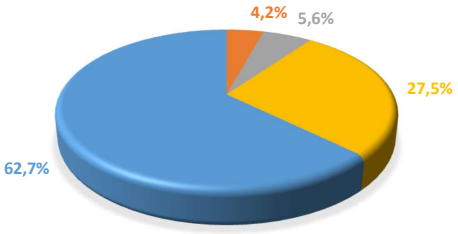
In the academic year 2019–2020, VPL was used only as a complement to the Mooshak, and no opinions were collected from students about its use.

Thus, regarding the survey carried out in 2018–2019, we present results on some issues that seem to be more important.

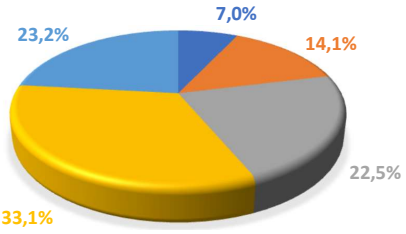
In the surveys, a five-level Likert scale was used (Figure 3).

Figure 3 Likert scale.

One of the questions intended to collect the students' opinion about the use of Distance Learning platforms such as, for example, Moodle. To this question, more than 90% of respondents answered that they totally or partially agree, with only 4.2% partially disagreeing, as can be seen in Figure 4a. It was also asked whether VPL was useful in the learning process. Most students (56.3%) agree (totally or partially) with the statement, with 21.1% of respondents saying that they disagree partially (14.1%) or totally (7.0%) (Figure 4b).



(a) Distance learning platforms (such as Moodle) are an added value.

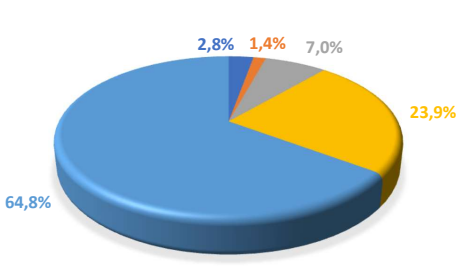


(b) Use VPL for solving programming exercises was helpful in my learning process.

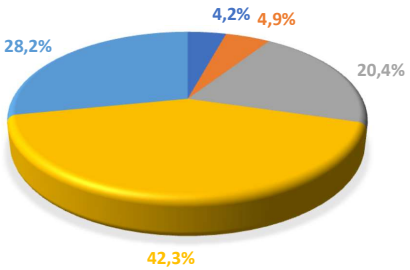
Figure 4 Survey responses about learning.

In the graphs presented in Figure 4, it is possible to observe that there is a great interest from students and a willingness to use technologies in the educational process. It is also possible to infer that most students believe that VPL was useful in their learning process.

With VPL it is possible to obtain an evaluation and, in case it does not correspond to the maximum value, the system presents the results of the failed tests, assisting the student to understand what needs to be corrected. As the automatic classification associated with the possibility of resubmission of works is one of the functionalities of VPL, it was intended to know the students' opinion about its usefulness, (Figure 5a), with 88.7% of the respondents saying that they totally or partially agree. It was also intended to gather the opinion on the added value of VPL for the teaching-learning process of programming. Only 9.1% of the students answered this question negatively, with 70.4% of the students totally or partially agreeing with the statement (Figure 5b).



(a) Be able to send assignments several times and get an automatic note is a very useful feature.



(b) VPL is an added value to the teaching-learning process of programming.

Figure 5 Survey responses about VPL usage.

From the analysis of the graphs presented in Figure 5, regarding the teaching-learning process, an extremely positive global appreciation by the students is evident, with a slight degree of disagreement regarding the statements of questions "VPL is an asset for the teaching-learning programming process" and "The use of VPL in solving exercises was an aid to my learning process". These are very rewarding results for the effort and dedication dedicated to the study and motivators for the future use of VPL in teaching programming.

3 Use of Mooshak in APROG

In addition to the study carried out with the VPL, we intend to test the potential of a new tool called Mooshak with which we have already had some contact and previous experience with programming contests.

3.1 The Mooshak application

Mooshak is a client–server application to fully manage and run programming contests [7]. It is a Web-based application and all of its functionalities are accessible by a Web-browser, and independent of Operating Systems. It is an open source system originally designed to manage online programming contests. The development of Mooshak was based on the rules of the ACM International Collegiate Programming Contest (ACM-ICPC) [8]. These contests are oriented to teams of students from higher education institutions from around the world [5]. Mooshak has been used in various programming contests such as SWERC (South Western Europe Regional ACM Programming Contest), MIUP (Inter-University Programming Marathon), TIUP (Inter-University Programming Tournament) or ToPAS (Tournament Programming Contest for Secondary Students). In the meantime, with the updates introduced, it is now possible to manage competitions from other fields and with different rules, namely, the Portuguese section of the International Olympiad in Informatics (IOI) [8].

A contest is a set of exercises students must solve using a programming language in a time window. Students submit source code for problem solving using a browser. Mooshak receives the source code and automatically compiles it, executes it using a set of pre-prepared tests and provides immediate feedback. The solution submitted to solve the problem is accepted if it satisfies all secret test cases prepared in advance for the exercise concerned. Otherwise, an error message will be displayed based on compilation / execution state.

Given the enhanced capabilities of automated code evaluation, Mooshak has been increasingly used as a pedagogical tool in teaching programming [9].

3.2 Mooshak – past experience

Since 2001, some programming competitions have been held among ISEP Computer Engineering students. Students loved the experience and were motivated to pursue other competitions. The best classifieds could participate in inter-university (MIUP) and international (SWERC) competitions.

Figure 6 illustrates a programming contest held at DEI-ISEP in 2006 and mediated with Mooshak.

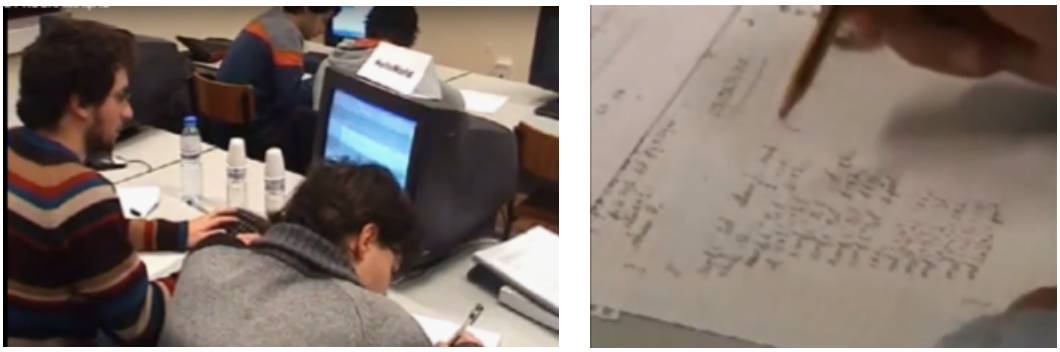


Figure 6 Programming contest on 2006 at ISEP.

The students consider that their participation in programming contests is an enriching personal experience and a relevant curriculum value. Given this environment of motivation and enthusiasm, it was considered interesting to replicate this experience in the classroom and try to foster this feeling for all students[6].

3.3 Contest in the classroom

Due to the background of the programming contests, it was intended to bring this motivation into the classroom and promote a kind of gamification in which students are valued for their performance in solving exercises and develop healthy competition among colleagues.

In addition, students have heterogeneous knowledge and distinct work speeds. These factors lead to the resolution of the problems proposed in class not being achieved at the same time by the students. An assessment tool gives students immediate feedback on their solutions.

Faced with large classes, the teacher is unable to give feedback to all students immediately after solving their problems. Waiting for teacher availability for resolution feedback can lead to a drop in student concentration and productivity. To minimize this concern, Mooshak can be used as the first line tool capable of giving immediate feedback to the student in solving the proposed exercises. When the teacher is available to the student, he or she can provide personalized feedback, not just in a particular exercise, but on all exercises already solved and validated in the first phase by Mooshak. Resolutions submitted and accepted by Mooshak are then reviewed by teachers to determine whether the answer is appropriate or, despite being validated by Mooshak has an inappropriate approach to the taught content. In this case the student is oriented to try a new resolution approach and the accepted submission may be revoked.

[6] Contest 2006 video available in `https://www.youtube.com/watch?v=xzfpmVey8BU`.

Each problem has a set of secret tests with an associated score. The submitted solution is graded with the sum of the successful test score. The submission score only reflects successful tests and not successful in all tests. The idea is that while the solution may only be a partial solution and not solve all scenarios, it is intended to enhance student effort and encourage them to look for a better solution for a better score.

The type of problems proposed try to follow the style of exercises of the international programming competitions (ACM-ICPC). A task is presented for the student to solve. One of the main concerns is to educate the student to respect what is requested and to be rigorous in their response. It is essential to respect strictly the format required in the job description. Any mistake, even slight, is enough for a solution not to be accepted. In all exercises, examples of input and their expected output are provided. This way, the student always has at least one example to test their program as intended.

The syllabus of the course is divided into three parts. For each part a contest is created with a set of exercises appropriate to that content and students are challenged to participate. The duration of the contest corresponds to the duration of the related program content.

In addition, Mooshak has some interesting features, namely, it provides a set of statistics that allow the student to track their performance and progress over time, as well as compare with peers through a ranking.

3.4 Results with Mooshak

At the end of the APROG course, 195 of the 304 students enrolled answered an anonymous survey to rate some questions on a scale of 1 to 5, where 1 means "strongly disagree" and 5 means "strongly agree". Table 3 summarizes the answers to the questions about using Mooshak.

It is observed that in general, the students positively appreciated the use of Mooshak. On the first question, 78% of students agreed, or strongly agreed, that automatic feedback was an asset in solving exercises. Also relevant is that 83% of students feel that using Mooshak motivated them to solve problems.

Table 3 Results of survey about using Mooshak.

	1	2	3	4	5
The use of Mooshak was very important as an aid in the feedback of the exercises	2%	5%	15%	41%	37%
The availability of Mooshak in class activities motivated you to be more active	2%	6%	10%	40%	43%

1 – Strongly Disagree; 2 – Disagree; 3 – Neutral; 4 – Agree; 5 – Strongly Agree

4 Conclusions and future work

Based on student responses to surveys, it is considered that the use of automatic assessment tools, and particularly Mooshak, has a considerable impact on students' motivation and active participation and can therefore significantly enhance the aid to teaching and learning programming.

One of DEI's strategic objectives is to continue new programming contest events.

In this sense, it is intended to consistently use a tool designed for programming contests to promote student training and increase their dexterity in solving competition style problems.

The experiments carried out with the VPL also demonstrated its great potential for use, with good acceptance by the students.

As VPL and Mooshak are two tools designed for different purposes, their use in APROG proved to be effective in a complementary perspective, since VPL was used after Mooshak and with more refined tests.

Nowadays, with a global job market, there is a growing trend in the use of automatic code validation tools, for the selection and recruitment of programmers. Thus, its use in academic and teaching contexts will represent an added value for students to enter the job market.

We believe that this study is opening doors to a new paradigm of programming teaching that will provide to involve student's motivation and new feedback mechanisms that will pass them the grant of a good performance giving them freedom to keep going.

References

1 Marina Umaschi Bers, Louise Flannery, Elizabeth R. Kazakoff, and Amanda Sullivan. Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, 72:145–157, 2014. doi:10.1016/j.compedu.2013.10.020.

2 Julio C Caiza and Jose M Del Alamo. Programming assignments automatic grading: review of tools and implementations. In *7th International Technology, Education and Development Conference (INTED2013)*, page 5691, 2013.

3 J.Marílio Cardoso, António Vieira de Castro, Rosa Barroso, Álvaro Rocha, and Rui Marques. Introducing vpl on a programming learning process. In *EDULEARN18 Proceedings*, 10th International Conference on Education and New Learning Technologies, pages 8499–8508. IATED, 2–4 july 2018. doi:10.21125/edulearn.2018.1981.

4 Eduarda Pinto Ferreira and Ângelo Martins. Eduscrum—the empowerment of students in engineering education? In *Proceedings of the The 12th International CDIO Conference*, page 596, Turku University of Applied Sciences, Turku, Finland, 2016.

5 ICPC. The icpc international collegiate programming contest. Accessed on 20.01.2020. URL: https://icpc.baylor.edu/.

6 Craig Larman. *Agile and iterative development: a manager's guide*. Agile software development series. Addison-Wesley Professional, Boston, USA, 2004.

7 José Paulo Leal and Fernando Silva. Mooshak: A web-based multi-site programming contest system. *Software: Practice and Experience*, 33(6):567–581, 2003.

8 José Paulo Leal and Fernando Silva. Using mooshak as a competitive learning tool. In *The 2008 Competitive Learning Symposium*, 2008.

9 Xiangfeng Luo, Marc Spaniol, Lizhe Wang, Qing Li, Wolfgang Nejdl, and Wu Zhang. *Advances in Web-Based Learning-ICWL 2010: 9th International Conference, Shanghai, China, December 8–10, 2010, Proceedings*, volume 6483. Springer, 2010.

10 João Maroco and Teresa Garcia-Marques. Qual a fiabilidade do alfa de cronbach? questões antigas e soluções modernas? *Laboratório de psicologia*, pages 65–90, 2006.

11 Juan Carlos Rodríguez-del Pino, Enrique Rubio Royo, and Zenón Hernández Figueroa. A virtual programming lab for moodle with automatic assessment and anti-plagiarism features. In *Proceedings of the 2012 International Conference on e-Learning, e-Business, Enterprise In-formation System*, 2012.

12 Jeff Sutherland. *Scrum: The Art of Doing Twice the Work in Half the Time*. Crown Business, 2014.

Game Elements, Motivation and Programming Learning: A Case Study

Davide R. Carneiro[1]
CIICESI, ESTG, Politécnico do Porto, Portugal
Algoritmi Center, Department of Informatics, University of Minho, Braga, Portugal
dcarneiro@estg.ipp.pt

Rui J. R. Silva
CETRAD, Centre for Transdisciplinary Development Studies,
University of Trás-os-Montes e Alto Douro, Vila Real, Portugal
rui.silva@utad.pt

Abstract

The learning of programming is traditionally challenging for students. However, this is also one of the most fundamental skills for any computer scientist, and is becoming an important skill in other areas of knowledge. In this paper we analyze the use of game-elements in a challenging long-term programming task, with students of the 3rd year of a Informatics Engineering degree. We conducted a quantitative study using the AMS scale to assess students' motivation. Results show that with the use of game-elements, students are both intrinsically and extrinsically motivated, and that they consider learning/working fun, which contributes positively to their academic performance.

2012 ACM Subject Classification Human-centered computing → Collaborative and social computing theory, concepts and paradigms

Keywords and phrases Motivation, Programming, Genetic Algorithms

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.5

Funding This work has been supported by national funds through FCT – Fundação para a Ciência e Tecnologia through project UIDB/04728/2020.

1 Introduction

Programming is arguably one of the most relevant skills that computer science students must acquire. With the advent and growing importance of computer science, students in other related fields (e.g. physics, biology, mathematics) are also expected to acquire at least some degree of programming/scripting skills. However, the teaching/learning of programming is admittedly and generally difficult.

In the specific case of learning programming at a higher-education level, some of the most common issues pointed out include the lack of previous knowledge on programming or related tasks, the difficulties in thinking in an abstract manner, the lack of time that must be divided with other demanding subjects, or the changes that occur in the student's life at different levels when they change from secondary to higher education [12]. These difficulties are even more expressive as in computer science courses programming subjects are generally taught starting at the first semester, when students are going through a significant change and still adapting.

Many authors have proposed different strategies, methodologies and theories of learning to try to address this issue and improve the teaching/learning process in the specific domain of Computer Science Education. Ben-Ari proposes Constructivism as a suitable theory

[1] Corresponding author

of learning for this domain, advocating that students do not passively absorb knowledge transmitted by the teacher or the book, but that they rather construct it building recursively on pre-existing knowledge, facts and beliefs [2]. This points towards a specific meaning for the term *teaching*. In this view, teaching becomes a task of helping or assisting someone to learn rather than simply presenting information. This also implies a shared responsibility: if students don't learn it is not (only) their fault.

In this context, Oliver proposes an interesting learning framework in this domain, constituted by three elements [10]. The first element is constituted by the "traditional" learning activities, which in this domain are generally designing/programming/testing tasks. The second element is constituted by the learning resources: materials that provide the content and context of the course, and help students construct their knowledge and meaning. Finally, the third element includes learning supports, which are elements that guide the student into constructing the necessary knowledge. These may range from the presence of the teacher (in a more traditional domain) to scaffolds such as automatic code generation, automatic diagram generation, or other software tools.

Other authors have also proposed specific tools and approaches to facilitate the teaching/learning process, and that can be seen as the learning supports proposed by [10]. Given the often abstract nature of programming tasks (and related/included activities such as designing an algorithm or a plan), visualization tools are often pointed out as an efficient aid in learning programming [7]. These may include program visualization, algorithm visualization or even visual programming tools [3]. Lister et al. propose the use of doodles in a rather informal setting, namely to assess the students' skills in reading and tracing code [8].

This paper describes a case-study in a computer science course in the academic year of 2018/2019. Specifically, we describe the use of gamification in the context of the *Artificial Intelligence* (AI) subject, in the Informatics Engineering degree, in the Higher School of Management and Technology of the Polytechnic Institute of Porto, in northern Portugal. This subject is taught in the second semester of the third year of the degree. Students are thus already expected to have prior experience in programming. However, and as described in Section 3, they must implement a Genetic Algorithm (GA) to solve a specific optimization problem. This is a rather new and abstract form of thinking on problem-solving for these students, and also a new form of programming, so the challenges previously pointed out remain.

Specifically, we describe the problem and the game-elements used with the main goal to motivate students to work in what would otherwise by a rather challenging and possibly dull task. We also describe the perceptions of the students regarding these game-elements and to what extent they contributed to their success and motivation during classes and while working in their assignment.

2 Motivation

Motivation is one of the key indicators for an individual to succeed in the learning process [6], leading the individual to apply her/his effort in order to achieve her/his goals. While motivation may emerge in different ways, it is usual to organize it into intrinsic (IMOT) and extrinsic (EMOT) motivation [14]. The former represents the individual desire to achieve something important. The latter is external, promoted by external factors.

An individual that is intrinsically motivated is one that gets involved in the learning process by the pleasure it gives her/him and because there is a sense of accomplishment. IMOT measures the extent to which an individual participates in a task for internal reasons

(e.g. curiosity, willingness to experience and overcome a given challenge) [11]. From the individual's perspective, it is a participation in which the task is an end in itself, intrinsically related to the individual's will. IMOT is composed by the Intrinsic Motivation to Know (IMTK), to Accomplish (IMTA), and to Atimulate (IMTS) [5].

On the other hand, an individual that is extrinsically motivated is one that will try to accomplish the easier tasks, while needing external impulses in order to feel motivated [9]. EMOT relates with the degree of participation of an individual in a task not by her/his own will but for external reasons such as rewards, competition with others, or performance-related reasons. EMOT is composed by four levels of growing degree of self-determination: Extrinsic Motivation External Regulation (EMER), Extrinsic Motivation Introjection (EMIN) and Extrinsic Motivation Identification (EMID) [4].

Usually, the motivation to learn comes from these two dimensions (IMOT and EMOT). However, it can also be affected by a third one: amotivation [13]. This concept (AMOT) was proposed by [5], and is related with a state of dismay, indifference, disinterest, self-discredit, prostration or depression [1]. AMOT represents a lack of interest or willingness in accomplishing a task or, on the other hand, results from a feeling of being unable or uninterested in reaching a goal. According to [5], it may result from frequent failure or negative feedback, leading the individual to assume that goals are not achievable.

3 The Learning Activity

As described in Section 1, in the 2018/2019 edition of Artificial Intelligence subject, the students had to program a Genetic Algorithm to solve a specific optimization problem. Given that this is a third year subject, students are expected to have prior experience in programming. However, the subject of AI is generally novel to them.

In this subject, three of the main paradigms of AI are addressed throughout the semester, namely: the Symbolist, the Evolutionary, and the Connectionist paradigms. Student assessment is done using two instruments: a practical assignment developed throughout the semester, and a written theoretical exam at the end of the semester. While going through these paradigms, students learn about Machine Learning (in its different forms), deduction, induction, and relevant fundamental algorithms such as Genetic Algorithms or Back Propagation.

The practical assignment is always dedicated to one of these three paradigms and aims to make the students devote themselves in depth to the chosen topic, guiding them into consolidating their knowledge through autonomous and practical implementation work, with the guidance of the teacher.

In the edition of 2018/2019, the practical assignment concerned the Evolutionary paradigm of AI. In that sense, they had to code, from scratch, a program to solve a specific optimization problem. In this case, the students had to program a Genetic Algorithm to allow an agent to learn to play the well known Super Mario Bros game. This game, developed and published by the popular video-game company Nintendo, was first published in the 80's and is still very popular nowadays, with new versions being created regularly. Virtually every current student played one or another version of this game and are thus familiar with their gameplay.

As detailed in Sections 3.1 and 3.2, students were provided with several learning supports and resources, to assist them in acquiring and/or constructing the necessary knowledge by the end of the semester.

This section describes the architecture implemented by the teacher to support the students in developing their work, and describes the game elements used to motivate them.

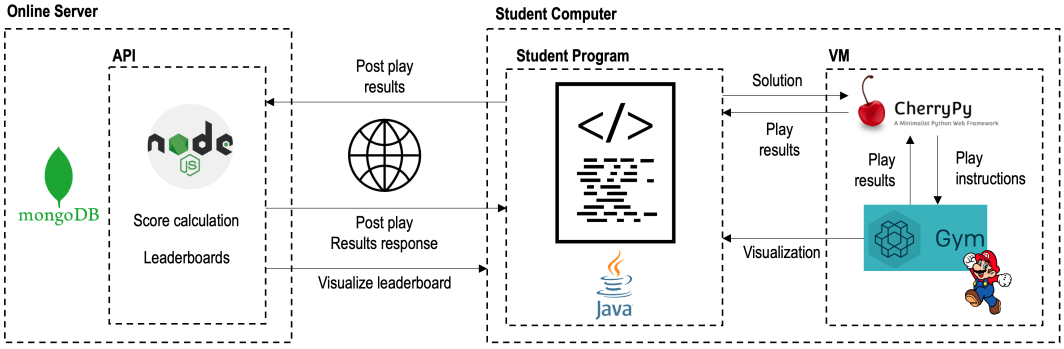


Figure 1 Architecture of the resources prepared for supporting the students to develop, test and validate their work.

3.1 Learning Supports and Resources

The task of programming a GA for a novice student may be challenging: it not only requires programming skills but also significant knowledge regarding evolutionary computation methods. The requirement of using a game emulator and integrating it in their application could further contribute negatively towards the learning goal.

Thus, asides from all the course contents available online, a group of learning supports was also provided to the students, in line with the constructivist view and Oliver's learning framework [2, 10]. These supports include two main elements (Figure 1): 1) an API for interacting with the game emulator and 2) an API for interacting with the Leaderboard.

The first API facilitates the interaction of the students' code with the game emulator. It is implemented in CherryPy and, among other functionalities, allows students to submit the solutions generated by their GA, and returns the result of running their solution. In this context, a solution is a set of game pad instructions starting in a given level. The response of the API includes elements such as the reason for loosing (if applicable), number of coins gathered, final level, number of points, among others. Students may also use this service to visualize their solution being played in the emulator, so as to better understand the practical effects of their decisions in the development of the GA. All these elements were provided in the form of a Virtual Machine (VM) that the students would run in their own computer.

The second API facilitates the interaction with the online Leaderboard. It includes services for students to post the results of their GA, whenever they feel like doing so. It also allows students to get the state of the Leaderboard, whether through the API or through a web page available online.

Both APIs were provided in the same programming language being used by the students to implement the GA (Java). Some examples of using the API were also provided, to facilitate the integration in their own code.

3.2 Game Elements

The importance of motivation in learning in general, and in learning programming in particular, has already been discussed in Section 2. This section details the game elements that were used in the learning activity in order to motivate students.

Given that this was a group work, students were first asked to constitute *teams*, rather than the traditional *work groups*. Each team was constituted by three members and had a name chosen by the members.

Students were then told that the practical assignment would be based on the well-known Super Mario Bros. video game, which was very well received. Students could participate in two different competitions. In the first competition students competed in a very specific level, attempting to reach the highest score in the minimum amount of time. In the second competition students started in the first level of the game and the goal was to get as far as possible, going through the several levels in sequence.

Asides from the use of the game, students were also provided an online Leaderboard that stimulated competition and motivation between teams. Teams would regularly post new top results of their GAs and motivate other teams to work further in order to gain access to the top of the board. Each entry in the Leaderboard included information about the team as well as about the solution (e.g. score, coins, enemies killed, level). The leaderborad included two pages, one for each type of competition.

In order to pass the practical assignment, students needed to participate in at least one competition. In order to score 18 or more values (out of 20) they needed to participate in both. Part of their score was attributed according to their position on the Leaderboard at the end of the competitions.

4 Methodology

A quantitative study was carried out, with data being collected through online questionnaires, using the AMS scale proposed by [14], adapted for students of Artificial Intelligence. Data was processed with SPSS v24, using several techniques such as Multiple Linear Regression (MLR), which allowed to test a model for measuring the motivation of students to study and learn. MLR allows to estimate the value of the dependent variable Intrinsic Motivation to Learn (IMTK) as a function of the independent variables EMER, EMIN, EMID, IMTA, IMTS and AMOT. The goal is to find the best relationship, and a statistically significant one, between the variables, to achieve the model that best explains motivation.

In order to evaluate the model, quality adjustment measures will be used such as Pearson's correlation coefficient, the coefficient of determination r^2, the adjusted r^2, the Variance Inflation Factor (VIF), and the Durbin-Watson Test.

The AMS was applied as an online questionnaire to 26 students of the 3rd year of the Degree on Informatics Engineering who where enrolled in the Artificial Intelligence subject. The questionnaire was applied after the conclusion of the subject.

A research model was tested for the population of the study. Its general expression is:

$$Y_i = \beta_0 + \beta_1 X_1 + \beta_1 X_1 + ... + \beta_k X_k + \varepsilon_i, i = 1, 2, ..., n$$

The proposed research model is given by:

$$IMTK = B_0 + B_1 AMOT + B_2 EMER + B_3 EMIN + B_4 EMID + B_5 IMTA + B_6 IMTS + \varepsilon$$

5 Results

The question that was given to the students in the questionnaire was phrased thus:

Why did you dedicate time studying for the Artificial Intelligence subject?

Table 1 Descriptive statistics for Amotivation.

	AMOT1	AMOT2	AMOT3	AMOT 4
Mean	1,19	1,03	1,19	2,39
Total	26 students ($\alpha = 0.847$)			

The answers of the students were given in a 7-point Likert scale, in which an average of 4 for a given sub-scale means that the statement moderately corresponds to the student's opinion. A value between 5 and 6 means that it corresponds significantly and a value of 7 that is corresponds totally.

Thus, for the EMOT and IMOT scales, a score equal or higher than 4 means that students are motivated. On the other hand, a high score in the AMOT scale means that students are less motivated.

5.1 AMOT

Generally, results show that students are motivated to study as the average values of amotivation are very close to the lower end of the scale (the closer to 1 the lower the amotivation) (Table 1).

This scale was composed by the following four questions:

- **AMOT1**: I honestly don't know, I feel like I am wasting my time studying AI.
- **AMOT2**: I don't see any point in attending AI classes and it does not interest me the least.
- **AMOT3**: I don't know. I don't understand what I'm doing in the AI classes.
- **AMOT4**: In the past I had good reasons to attend AI classes, now I wonder whether I should continue.

A global analysis allows to conclude that the large majority of students are motivated to study AI. However, in what concerns question AMOT4, the value is closer to 3, which may point to a certain tendency to amotivation related with the initial expectations towards the subject. This calls for measures and strategies to minimize this tendency in future editions. However, it must also be noted that the questionnaire was administered after the conclusion of the subject, that is, after the students knew the results of the evaluation moments. This result may thus be partly influenced by students who did not achieve the expected marks.

5.2 Extrinsic Motivation

In the EMOT scale, the higher the values, the higher the motivation. Three sub-scales were analyzed: EMER (with an average of 5.3), EMIN (4.69) and EMID (5.98). This points out that students are generally extrinsically motivated to study (Table 2).

The EMER sub-scale was composed by the following questions:

- **EMER1**: Because I need a degree to get a better job in the future and the AI subject is mandatory in the Informatics Engineering degree.
- **EMER2**: In order to obtain a more prestigious job in the future.
- **EMER3**: Because I want to have a "good life" in the future.
- **EMER4**: To have a better salary in the future.

EMER4, which is related with the prospect of a better salary in the future, is the question that scores the lowest, which indicates that money is by itself not a good enough motivator, or that students do not associate AI skills with better salaries.

Table 2 Descriptive statistics for Extrinsic Motivation.

	EMER1	EMER2	EMER3	EMER4	EMIN1	EMIN2	EMIN3	EMIN4	EMID1	EMID2	EMID3	EMID4
Mean	5,38	5,61	5,38	4,84	4,69	4,69	4,70	4,68	6,23	5,46	5,88	6,34
Total	$\alpha = 0.831$				$\alpha = 0.994$				$\alpha = 0.867$			

Concerning the EMIN sub-scale, it was composed by the following questions:

- **EMIN1**: Because when I succeed in any assignment related to AI I feel important.
- **EMIN2**: In order to prove myself that I can pass the AI subject.
- **EMIN3**: To prove myself I am an intelligent person.
- **EMIN4**: Because I want to prove myself that I am an intelligent person.

The analysis of the responses allows to understand that this dimension, although scoring above 4.5, is the one with the lowest score concerning external motivation. This points out that from the external motivators, the ones related with the self are the less important.

Finally, the EMID sub-scale included the following questions:

- **EMID1**: Because I believe that the AI subject will prepare me better for my future career.
- **EMID2**: Because eventually, what I learn in the AI subject will allow me to find a job in a field that I like.
- **EMID3**: Because I believe that AI knowledge will improve my skills as a worker.
- **EMID4**: Because the topics addressed in the AI subject will allow me to make better career choices.

This sub-scale is the one with the highest scores, with values around 6. It is interesting to note that the prospect of a better salary (EMER4) does not motivate the students as much as the prospect of a better job. This may point out that students are oriented towards satisfying jobs, in line with their fields of interest, rather than money. Moreover, they also believe that AI skills will provide them with better options in the future. This is clear in the highest-scoring question EMID4.

5.3 Intrinsic Motivation

Concerning Intrinsic Motivation, the values for the three sub-scales IMTK, IMTA and IMTS are, respectively, 6.16, 5.4 and 5.83. This shows that students generally study AI for the pleasure it gives them and by their own will (Table 3).

Concerning the IMTK scale, it was composed by the following questions:

- **IMTK1**: For the pleasure I feel while overcoming my own limits while learning AI.
- **IMTK2**: For the pleasure I feel while solving academic assignments in the field of AI.
- **IMTK3**: For the pleasure I feel when I overachieve in my personal achievements.
- **IMTK4**: Because the AI subject allows me to experience personal satisfaction on my path towards academic excellency.

The results on the IMTK1 and IMTK2 questions show that students feel significant pleasure and satisfaction while working in this subject, which contributes very positively to their engagement.

Table 3 Descriptive statistics for Intrinsic Motivation.

	IMTK1	IMTK2	IMTK3	IMTK4	IMTA1	IMTA2	IMTA3	IMTA4	IMTS1	IMTS2	IMTS3	IMTS4
Mean	6,46	6,30	5,88	6,01	5,57	5,30	5,61	5,11	6,11	6,01	5,73	5,46
Total		$\alpha = 0.837$				$\alpha = 0.777$				$\alpha = 0.806$		

Concerning IMTA, the following questions were included:

- **IMTA1**: Because I feel pleasure and satisfaction when learning new topics in the field of AI.
- **IMTA2**: For the pleasure I feel when I learn new things.
- **IMTA3**: For the pleasure I feel when deepening my knowledge on topics that I like in AI.
- **IMTA4**: Because the AI subject allows me to learn about topics that I'm interested in.

Although IMTA scored, on average, above 5, it is also the one with the lowest score. IMTA4 was the question with the lower score, which may point out that students may not be so interested in certain topics taught during the subject.

Finally, in what concerns the IMTS, the following questions were considered:

- **IMTS1**: Because I really like to attend AI classes.
- **IMTS2**: Because, for me, AI classes are fun.
- **IMTS3**: For the pleasure I feel when I participate in discussions about AI with interesting teachers.
- **IMTS4**: For the good feelings I experience when I read about AI.

All questions in the IMTS sub-scale were scored with values very close to 6. The two higher-scoring questions were IMTS1 and IMTS2, which show that students like and have fun in classes. This is very likely due to the use of games in classes, which make for a relaxed and fun environment.

5.4 Proposed Model

In order to explain in a more robust way the data, they were analysed through a MLR, selecting statistically significant variables that would provide a thorough model.

We analyzed the assumptions of the model, namely the ones of normal distribution, homogeneity and error independence. The first two assumptions were validated graphically and the third was validated with the Durbin-Watson statistic. The VIF was also used to diagnose the multicolinearity, eliminating variables with strong colinearity. Table 4 details the results of the model tested.

The final model was thus:

$$IMTK = 0.173 + 0.076EMIN + 0.309EMID + 0.460IMTA + 0.498IMTS$$

The final analysis of the obtained model shows a model with $R^2a = 0.810$, with a high explanatory power since the dimensions that compose it explain a large proportion of the IMTK. The dimensions AMOT and EMER did not influence students' IMTK, which shows that their amotivation is reduced. Concerning the fact that EMER is not statistically significant, this show that students are not influenced by external pressures to execute study tasks, not fearing possible punishment and not valuing the rewards obtained with the reaalization of the activity.

Table 4 Linear regression model.

Dependent Variable: IMTK				
Dimensions	Initial Model		Final Model	
	B	t	*B*	t
Constant	-.017	-.059	.173	.745
AMOT	.088	1.091	.088	1.091
EMER	-.030	-.504	-.030	-2.296
EMIN	-.069	-1.722	.076**	4.367
EMID	.262***	4.450	.309***	5.766
IMTA	.331***	5.785	.460***	11.918
IMTS	.491***	11.555	.498***	.745
VIF	[1.206 – 2.842]		[1.634 – 2.721]	
R	.903		.902	
R^2	.816		.814	
R^2a	.810		.810	
Durbin-Watson	1.771		1.750	

** $\rho < 0.05$ *** $\rho < 0.001$

6 Conclusions

This study focused on the perceptions of 3rd year students of the Informatics Engineering degree, in the AI subject. The study allowed to understand the motivation of 26 students, and more specifically what dimensions of motivation influence their willingness to study. The major limitation of the study is the relatively small size of the population. The main reason for this was that the questionnaire was sent to students *a posteriori*, after they knew the results of their score in the subject. However, most of the students were already not attending school and probably not using their institutional e-mail, which was used to contact them. Still, results are interesting and point out to a positive effect of the use of game elements in student motivation.

Several dimensions and sub-scales of motivation were evaluated. Generally, students show positive values of extrinsic and intrinsic motivation towards study. However, there is also evidence of small groups of students who are less motivated to study.

There is evidence that students are more motivated intrinsically than extrinsically, showing that they are more self-motivated than being pressured by external influences. Levels of motivation increase along the Ryan & Deci's Self-Determination Continuum [4], with IMOT dimensions having a higher relevance than EMOT's.

The building of the model to estimate the structure of IMTK shows that some variables of the initial model were not statistically significant. A new model was tested, removing the dimensions that had not been validated, originating a more robust and significant model, that explains 81% of IMTK.

However, in this model, the AMOT and EMER dimensions were removed, which were no longer significant for IMTK. Concerning EMOT, the EMID dimension ($\beta = 0.309; \rho < 0.001$) was the more important for the construction of students' extrinsic motivation, followed by EMIN ($\beta = 0.076; \rho < 0.05$). Concerning the IMOT dimension, both IMTS ($\beta = 0.498; \rho < 0.001$) and IMTA ($\beta = 0.460; \rho < 0.001$) were relevant for motivation, with a sligh advantage of IMTS.

In conclusion, this study shows that this group of students, who used the previously described game-elements in a complex learning task, saw their motivation positively influenced. Maybe the use of a non-traditional way of studying and working contributes to what may, eventually, make the difference in their academic performance when learning AI. Thus, if the approach considered also facilitates learning, this kind of approaches should be further considered in the future as a facilitator of learning in this and related fields of knowledge.

References

1 Vassilis Barkoukis, Haralambos Tsorbatzoudis, George Grouios, and Georgios Sideridis. The assessment of intrinsic and extrinsic motivation and amotivation: Validity and reliability of the greek version of the academic motivation scale. *Assessment in Education: Principles, Policy & Practice*, 15(1):39–55, 2008.

2 Mordechai Ben-Ari. Constructivism in computer science education. In *Acm sigcse bulletin*, volume 30 (1), pages 257–261. ACM, 1998.

3 Eileen Costelloe. Teaching programming the state of the art. *The Center for Research in IT in Education. Dept. of Computer Science Education. Dublin: Trinity College.*, 2004.

4 Edward L Deci, Richard Koestner, and Richard M Ryan. Extrinsic rewards and intrinsic motivation in education: Reconsidered once again. *Review of educational research*, 71(1):1–27, 2001.

5 Edward L Deci and Richard M Ryan. The general causality orientations scale: Self-determination in personality. *Journal of research in personality*, 19(2):109–134, 1985.

6 John Hattie. *Visible learning: A synthesis of over 800 meta-analyses relating to achievement*. routledge, 2008.

7 CD Hundhausen. A meta-study of software visualization effectiveness. *Unpublished comprehensive examination paper, Department of Computer Science, University of Oregon, Eugene, OR.*, 1997.

8 Raymond Lister, Elizabeth S Adams, Sue Fitzgerald, William Fone, John Hamer, Morten Lindholm, Robert McCartney, Jan Erik Moström, Kate Sanders, Otto Seppälä, et al. A multi-national study of reading and tracing skills in novice programmers. In *ACM SIGCSE Bulletin*, volume 36 (4), pages 119–150. ACM, 2004.

9 Alf Lizzio, Keithia Wilson, and Roland Simons. University students' perceptions of the learning environment and academic outcomes: implications for theory and practice. *Studies in Higher education*, 27(1):27–52, 2002.

10 Ron Oliver. Exploring strategies for online teaching and learning. *Distance Education*, 20(2):240–254, 1999.

11 César Orsini, P Evans, V Binnie, P Ledezma, and F Fuentes. Encouraging intrinsic motivation in the clinical setting: teachers' perspectives from the self-determination theory. *European Journal of Dental Education*, 20(2):102–111, 2016.

12 Danijel Radošević, Tihomir Orehovački, and Alen Lovrenčić. Verificator: educational tool for learning programming. *Informatics in Education*, 8(2):261–280, 2009.

13 Jennifer Christie Siemens, Scott Smith, Dan Fisher, Anastasia Thyroff, and Ginger Killian. Level up! the role of progress feedback type for encouraging intrinsic motivation and positive brand attitudes in public versus private gaming contexts. *Journal of interactive marketing*, 32:1–12, 2015.

14 Robert J Vallerand, Luc G Pelletier, Marc R Blais, Nathalie M Briere, Caroline Senecal, and Evelyne F Vallieres. The academic motivation scale: A measure of intrinsic, extrinsic, and amotivation in education. *Educational and psychological measurement*, 52(4):1003–1017, 1992.

An Augmented Reality Mathematics Serious Game

José Manuel Cerqueira
Polytechnic Institute of Cávado and Ave, Barcelos, Portugal
cerqueirajm@gmail.com

João Martinho Moura
Polytechnic Institute of Cávado and Ave, Barcelos, Portugal
jmoura@ipca.pt

Cristina Sylla
Research Centre on Child Studies (CIEC), Universidade do Minho, Braga, Portugal
cristina.sylla@ie.uminho.pt

Luís Ferreira
2Ai – Applied Artificial Intelligence Lab, Polytechnic Institute of Cávado and Ave, Barcelos, Portugal
lufer@ipca.pt

Abstract

This article presents the results obtained from an experiment using an Augmented Reality (AR) serious game for learning mathematical functions in middle school, in contexts that resort to Game Based Learning. A serious game was created specifically for this purpose and allowed to conduct an exploratory study with a quantitative and qualitative methodological approach, with two groups of teachers of different subjects: mathematics and informatics. The game, called FootMath, allows the visualization, manipulation and exploration of linear, quadratic, exponential and trigonometric mathematical functions, through the simulation of a 3D football game, in which the user can change the function parameters with different values, in order to score a goal. It was tested the potential use of AR technologies in learning scenarios, considering the teacher's perspective. According to the findings, FootMath was considered to be a promising and innovative tool to be incorporated in real mathematics teaching scenarios.

2012 ACM Subject Classification Computing methodologies → Mixed / augmented reality

Keywords and phrases Serious Game, Augmented Reality, Mathematics, Functions

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.6

Acknowledgements 2Ai – Applied Artificial Intelligence Lab, Polytechnic Institute of Cávado and Ave, Portugal

1 Introduction

Although still in early stages, research has shown that Augmented Reality (AR) has become increasingly popular [9] as a promising innovative technology that offers a new way to blend virtuality and reality [5]. The number of mobile applications with AR targeting multiple areas of education has rapidly increased [17, 21] and it is thriving. The interest in AR is not limited to education but encompasses other areas of knowledge such as science, engineering, business and entertainment. To some extent, this growth results from the adaptability of AR to the technological changes of mobile platforms [14], such as smartphones and tablets (usually equipped with a camera, gyroscope and accelerometer) as well as the increase of the Internet speed, local networks, and ubiquitous access. The fast pace of technological changes has made AR-based resources appealing to different agents of education due to their potential and the new pedagogical opportunities [26], particularly on account of their level of interaction, and

how the information is exchanged, the spatial integration of the surrounding environment, and the possibility of connecting with one's daily life. While facing this new paradigm, new tools and pedagogical strategies that allow students to visualize the world in a new way, are imperative. AR technology can be experienced through earphones, glasses, and headsets or Head Mounted Displays (HMD). The Microsoft Hololens and the Magic Leap are examples of AR projects based on HMD [9]. However, AR is mostly applied in mobile devices such as tablets and smartphones. In the future, other (wearable) devices such as smartwatches and contact lenses could possibly be other promising targets of AR technology. Concerning mobile devices, the virtual objects are included in the image that is given by the device's camera (lens), in real-time, while realistically blending with the tangible world. In this context, we set out to promote new spaces for learning while considering the challenges students face when learning mathematics. This study was carried out with the aim of understanding and answering the following Research Questions: - RQ1. What are the challenges that AR poses when learning basic mathematical functions in a digital game? - RQ2. What do teachers think about this new learning opportunity based on a digital AR game? The investigation revolved around an exploratory study on the impact and challenges of using (i) a serious game for learning mathematical functions, and (ii) a component of tangible interaction that is mediated by AR technology for a mobile platform, such as Android or iOS. The choice of these platforms stems from the fact that the majority of middle school students have a cell phone and/or tablet (sometimes equipped with superior technology than the resources that are available in schools). Through a playful (gameplay) approach, the game intends to help problem solving, and the understanding of some mathematical concepts associated with basic functions, such as $y = ax + b$ (linear function) and $y = ax^2 + bx + c$ (quadratic function). The game environment intends to simplify the processes that are related to both teaching and learning. Currently, the use of serious digital games and AR in education are two important research topics [16, 23]. Research on Game Based Learning (GBL) is by far more comprehensive than research on AR in education, since GBL and its associated technologies are older than the use of AR in education. A search of the terms "Game Based Learning" on Google Scholar, provides 3,5 million results; whereas the terms "Augmented Reality", provided around 700.000 results. Research related to the improvement of the quality of education refers that the adequate integration of technological resources with a particular pedagogy and contents, promotes good practices and the success of learning and teaching experiences [15, 24]. In a brief overview, the literature related to the teaching and learning of mathematics describes positive aspects and multiple advantages of using digital technologies, in particular in face-to-face lessons with the different methods of teaching and learning. The world's largest organization of teachers of mathematics, the National Council of Teachers of Mathematics, with members and affiliates in the United States and Canada, provides a method for the introduction of technology in mathematics lessons, anchored on the notion that technology has the potential to improve the learning of mathematics, to support more effective teaching of mathematics, and to significantly influence what mathematics can teach [20]. In this context and aligned with the challenges of developing tools and methodologies that motivate the students, this work aimed to develop and investigate the potential of an AR serious game for learning mathematical functions in e-contexts. The following statement is particularly relevant.

> *"(. . .) Many children find mathematics difficult and boring. But they are curious, and they love to have fun with exciting things around them. Appropriate activities can be found to stimulate them to have fun and love learning mathematics (. . .)"* [27, p. 759]

2 Background and related work

When it comes to learning, one cannot question the value and relevance of digital games: gaming is instinctive, and, by playing, one is rewarded with a learning experience [19]. Marc Prensky lists several characteristics that make digital games so engaging and appealing to millions of people: games are a type of entertainment, that is interactive and adaptable. Games have goals, results, feedback, rewards, conflict, competition, challenges, opposition, problems, representation and history [22].

2.1 Learning based on games

Educational or serious games can be defined as digital games to improve and promote learning, with the advantage of being a type of entertainment [16]. These types of games (video or digital) seem to increase the player's motivation while allowing the progression and assimilation of new learning contents within a continuous and significant narrative. Malone and Lepper [18] identified four strengths that might potentially promote a learning environment as a game activity, which is intrinsically motivational: fantasy, curiosity, challenge, and control. Thereby, with these four elements, Malone and Lepper show what educational games should enforce, while attempting to define specific principles for the design of these same games: 1) The games should use fantasy to reinforce the learning outcomes and stimulate the pre-acquired interests of the student; 2) Games must create sensorial stimuli (such as through audio-visual media) and develop the cognitive curiosity of the student; 3) Games should pose a challenge; through the achievement of goals and feedback, the student must feel continuously stimulated, and the difficulties must be increased taking into account a balance between the obstacles that are posed by the game and the acquisition of skills, in order to prevent that the student gets bored or frustrated; and 4) The student must feel a sense of control through the feedback that is provided throughout the game; i.e., s/he should feel that the learning outcomes are determined by his/her own actions. These four key elements may potentially enhance learning and must be taken into consideration in order to understand if a game meets what is required of it and if it is adequate to be used as a learning resource [18]. According to the literature, digital games may be successfully used as complementary learning tools [6]. Furthermore, the use of digital games in learning encourages the research of new paradigms of teaching-learning given their advantages over traditional learning materials, such as encouraging students to make decisions and the experimentation of different solutions to solve problems [10]. As previously mentioned, the new technological developments have the potential to create innovative scenarios for learning and teaching in real and virtual environments, as well as those that combine them. In particular, the use of AR digital games in the classroom allows the introduction of different methodologies and learning strategies that are focused on the student/group, while exploring the motivation and attention that is incited by technology (interactive digital environment). In mathematics lessons, educational digital games potentiate immersive experiences (in different degrees) in which the players may retain information, think and solve problems in an amusing way. Therefore, the games that are used in educational contexts involve pedagogical aspects. However, and although the pedagogical elements must be taken into consideration, the element of entertainment must be brought to the forefront. It is precisely the educational component that transforms pure entertainment into a powerful learning tool referred to as serious games. Serious games recur to pedagogy to introduce instruction in the gaming experience [28].

2.2 Augmented Reality in Education

The areas of applicability of Virtual Reality, Augmented Reality and Mixed Reality technologies are increasingly more diverse. These types of technologies have been gaining interest and notoriety within the research community. The new AR technology, in particular has appeared in education and in research work, showing that its application may have rather positive learning outcomes [8, 11]. The present research intends to analyze the value and the impact of AR technology in the process of teaching and learning, while presenting AR as part of a new paradigm of education. AR applications are particularly adequate to the visualization of space, offering advantages over other forms of materials (such as books), because they allow to simplify the visualization of 3D objects, by simulating dynamic processes that are not easily visible in real life [7, 12]. Consequently, AR applications contribute to a reduction of the cognitive load of the users [3, 13], by freeing cognitive resources. A systematic review of research work on the use of AR in educational contexts [1] points out motivation, interaction, collaboration, and learning as the main advantages of such use. These studies suggest that AR can be used effectively in educational contexts, while contributing to enhancing the motivation and collaboration of the student, potentially resulting in a better process of learning. Billinghusrt and Duenser [2] state that:

> *"(. . .) AR educational media could be a valuable and engaging addition to classroom education and overcome some of the limitations of text-based methods, allowing students to absorb the material according to their preferred learning style (. . .)".*

The pedagogical approach to be adopted while facing an AR resource and the alignment between the design/interface of the technology, the methodology of teaching and the experiences of learning are key elements that must be taken into consideration [25]. Although AR offers new opportunities for learning, it also poses new challenges, both for teachers and students.

3 Technical Development of FootMath

The FootMath is an AR serious game, following the Game Based Learning strategy (Fig. 1), was developed to run on Android mobile devices. In order to do so, two development platforms were used: the game engine Unity[1] and the augmented reality engine Vuforia[2] platform. FootMath was designed to incorporate AR and to work with 2D physical markers (Fig. 2).

In its execution, the markers are a part of the tangible interface and are used to activate the exploration of the different mathematical functions. Previous experiments with physical AR markers allowed us to explore spatial and mathematical concepts with significant results in the learning process [4]. The database of the physical markers was created in the Vuforia platform and then printed in paper. Vuforia indicated the quality and the degree of precision of each marker, by classifying them according to a particular scale. In order for the physical markers to be successfully detected, each marker must rank high on the scale when it comes to their visual characteristics.

[1] `https://unity.com/`
[2] `https://developer.vuforia.com/`

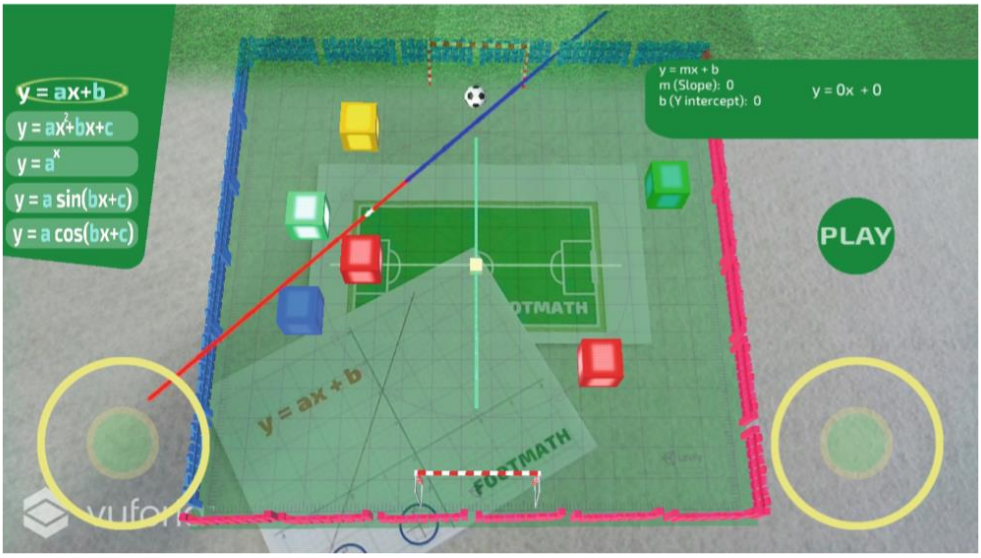


Figure 1 FootMath.

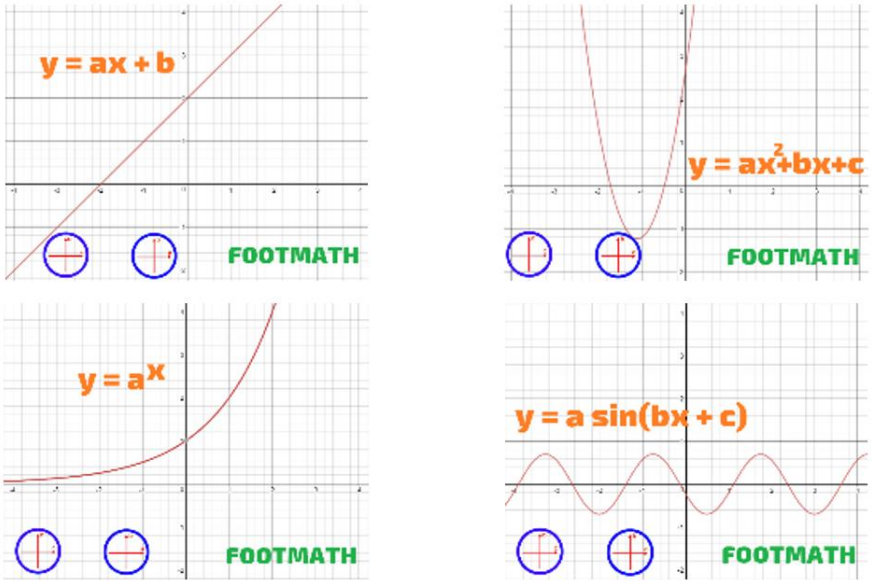


Figure 2 (clockwise) Markers for the functions: linear,$y = ax + b$; quadratic,$y = ax^2 + bx + c$, exponential, $y = a^x$ and trigonometric sine, $y = asin(bx + c)$.

4 Exploratory activity with FootMath

In the present study, our sample size consisted of 22 middle school teachers, divided into two groups (created in different dates and places): one group of 11 teachers of mathematics and another group of 11 teachers of informatics. After experimenting with FootMath, the teachers had to fill a questionnaire that was divided into three parts: i) the first part revolved around the profile design of the teacher, including questions regarding their teaching experience, the levels of education they teach, if they had used AR applications and if they had any

type of experience with AR games or programs to teach; ii) the second part was designed to understand the perspective of the teachers on the trial of FootMath; and iii) the third part was created to collect the opinions of the inquired on the challenges that AR poses when applied to educational purposes, particularly in mathematics. 17 of the 22 teachers were female (77,3%). The direct observation of the participants on the execution of the proposed tasks (during the trial and group interviews) and the listening to the groups, were the tools used to collect and fill the grid of results. Some examples of the proposed tasks were: placement and shift of the physical markers (targets), with the use of joysticks (manipulation of the parameters of the functions when looking for the zero of the function); experimenting with changing the slope (positive/negative) of the linear function ($y = ax + b$); attempt to score goals with the linear function in an early phase and, afterward, with the quadratic function ($y = ax^2 + bx + c$). Thus, it was possible to notice that FootMath sparked the interest and the engagement of every single participant.

4.1 Analysis and discussion of the obtained data

Through the experimentation and exploration of FootMath, teachers of different subjects (mathematics and informatics) focused on pointing out the benefits and challenges related to the potential of AR serious games with basic mathematical functions. The view of the agents of teaching was taken into consideration regarding the potential of using this resource as a playful educational tool, capable of motivating and involving the students in problem-solving and logical thinking. After the experimentation and analysis of FootMath, every math teacher agreed on the following aspects: a) it was an interactive and innovative experience; b) if applied in a classroom as a strategic resource, the game can contribute to certain advantages when it comes to attention, interaction, experimentation and engagement; c) the game contributes to an innovative and adjusted observation and exploration of functions such as: linear ($y = ax + b$), quadratic ($y = ax^2 + bx + c$), trigonometric (sine and cosine) and exponential ($y = a^x$); d) AR can be used inside the classroom as a complement to the processes of learning and teaching of items related to basic mathematical functions; e) AR might increase the level of interest of the students for mathematics as a whole; f) it is valuable to use books and interactive worksheets with AR; g) overall, it can increase/enhance the process of learning. Similar results were found for computer science teachers. The teachers of both groups (22) expressed their opinion (agree or completely agree) as follows: a) Game experience – innovative (95%), interesting (91%), motivating (82%), entertaining (91%), dynamic (86%), interactive (95%), and fluid (68%); b) Advantages of using FootMath inside the classroom – attention (100%), interaction (95%), commitment (95%), experimentation (100%), engagement (95%) and learning (86%); c) FootMath might provide a complementary strategy to the formal approaches to the concepts of linear functions ($y = ax+b$) and quadratic functions ($y = ax^2 + bx + c$) on the levels of knowledge (91%), understanding (82%) and with the application (86%); d) The game contributes to an innovative and adjusted observation and exploration of functions such as: linear ($y = ax + b$), quadratic ($y = ax^2 + bx + c$), trigonometric (sine and cosine) and exponential ($y = a^x$); e) The incorporation of the graphs of the basic functions in a 3D world (football field) – correct (86%), feasible (82%) and viable (82%); f) AR increases the level of interest of the students for mathematics as a whole. In a nutshell, the results that were obtained from the questionnaire confirm a positive impact on the possible use of FootMath inside classrooms. This new opportunity of learning based on a digital AR game presents several benefits, and it might also improve the learning process of basic mathematical functions.

5 Conclusion

The present study focused on evaluating the use of FootMath, an AR serious game, as a complementary tool in an educational setting. The primary goal of this virtual game is to simplify the visualization and understanding of linear, quadratic, exponential, and trigonometric (sine and cosine) functions by students and also of enhancing their interest in these subjects. It was possible to test and evaluate FootMath through the evaluation of teachers with several years of teaching experience and, therefore, specialists in the subjects that FootMath deals with. The research allowed us to explore the benefits and the challenges related to the potential of AR serious games for learning basic mathematical functions. In conclusion, FootMath is a promising tool that offers students a different way to learn math by presenting information in a different format and allowing innovative interaction. It is a complimentary resource to motivate and engage students in learning with playing.

5.1 Future work

In the future, other mathematical functions at different levels of difficulty will be developed. Furthermore, there is a plan to expand the features of this serious game by replacing the physical markers by Google's ARCore technologies in order to explore and manipulate the functions in a more interactive way with the surrounding environment and different angles and distances. Further, there is an intention to explore the application of AR technologies in the Student-Centered Learning of Mathematics, integrated with Artificial Intelligence tools to achieve the ability to interpret student behavior and dynamically adjust or reconfigure their teaching process.

References

1. Jorge Bacca, Silvia Baldiris, Ramon Fabregat, Sabine Graf, and Kinshuk. Augmented reality trends in education: A systematic review of research and applications. *Educational Technology and Society*, 17(4):133–149, 2014.
2. Mark Billinghurst and Andreas Duenser. Augmented Reality in the Classroom. *Computer*, 45:56–63, 2012. doi:10.1109/MC.2012.111.
3. Matt Bower, Cathie Howe, Nerida McCredie, Austin Robinson, and David Grover. Augmented reality in Education — Cases, places, and potentials. *Educational Media International*, 51, 2014. doi:10.1080/09523987.2014.889400.
4. José Cerqueira, Bárbara Cleto, João Martinho Moura, and Cristina Sylla. Visualizing platonic solids with augmented reality. In *Proceedings of the 17th ACM Conference on Interaction Design and Children - IDC '18*, pages 489–492, New York, New York, USA, 2018. ACM Press. doi:10.1145/3202185.3210761.
5. Yunqiang Chen, Qing Wang, Hong Chen, Xiaoyu Song, Hui Tang, and Mengxiao Tian. An overview of augmented reality technology. *Journal of Physics: Conference Series*, 1237:022082, 2019. doi:10.1088/1742-6596/1237/2/022082.
6. Athanasios S. Drigas and Marios A. Pappas. On line and other game-based learning for mathematics. *International Journal of Online Engineering*, 11(4):62–67, 2015. doi:10.3991/ijoe.v11i4.4742.
7. Andreas Duenser, Lawrence Walker, Heather Horner, and Daniel Bentall. Creating interactive physics education books with augmented reality. In *Proceedings of the 24th Australian Computer-Human Interaction Conference*, pages 107–114, 2012. doi:10.1145/2414536.2414554.
8. Anne Estapa and Larysa Nadolny. The Effect of an Augmented Reality Enhanced Mathematics Lesson on Student Achievement and Motivation. *Journal of STEM Education*, 16(3):40–49, 2015.

9 Enrico Gandolfi. Virtual Reality and Augmented Reality. In R.E. Kennedy, K, Ferdig, editor, *Handbook of Research on K-12 Online and Blended Learning (2nd ed.)*, pages 545–561. ETC, 2018. doi:10.1007/978-3-319-98213-7_20.

10 Mark Griffiths. The educational benefits of videogames. *Education and health*, 20(3):47–51, 2002.

11 Adrian Iftene and Diana Trandabăt. Enhancing the Attractiveness of Learning through Augmented Reality. *Procedia Computer Science*, 126:166–175, 2018. doi:10.1016/j.procS.2018.07.220.

12 Hannes Kaufmann. The potential of augmented reality in dynamic geometry education. In *12th International Conference On Geometry and Graphics (ISGG), Ago*, pages 6–10, 2006.

13 Mehmet Kesim and Yasin Ozarslan. Augmented reality in education: current technologies and the potential for education. *Procedia-Social and Behavioral Sciences*, 47:297–302, 2012.

14 Tasneem Khan, Kevin Johnston, and Jacques Ophoff. The Impact of an Augmented Reality Application on Learning Motivation of Students. *Advances in Human-Computer Interaction*, 2019, 2019. doi:10.1155/2019/7208494.

15 Babak Khoshnevisan and Nhu Le. Augmented Reality in Language Education: A Systematic Literature Review. In *Global Conference on Education and Research*, 2019.

16 Jingya Li, Erik Spek, Loe Feijs, Feng Wang, and Jun Hu. Augmented Reality Games for Learning: A Literature Review. In *Distributed, Ambient and Pervasive Interactions*, pages 612–626, 2017. doi:10.1007/978-3-319-58697-7_46.

17 Tzung-Jin Lin, Henry Been-Lirn Duh, Nai Li, Hung-Yuan Wang, and Chin-Chung Tsai. An investigation of learners' collaborative knowledge construction performances and behavior patterns in an augmented reality simulation system. *Computers & Education*, 68:314–321, 2013.

18 T. Malone and M. Lepper. Intrinsic motivation and instructional effectiveness in computer-based education. In *Conative and Affective Process Analyses*, pages 255–286. Hillsdale NJ: Erlbaum, 1987.

19 Carlos Martinho, Pedro Santos, and Rui Prada. *Design e Desenvolvimento de Jogos*. Tecnologias de Informação. FCA, 2014.

20 NCTM. Strategic Use of Technology in Teaching and Learning Mathematics, 2015. URL: https://www.nctm.org/Standards-and-Positions/Position-Statements/Strategic-Use-of-Technology-in-Teaching-and-Learning-Mathematics/.

21 Danakorn Nincarean, Mohamad Bilal Alia, Noor Dayana Abdul Halim, and Mohd Hishamuddin Abdul Rahman. Mobile Augmented Reality: The Potential for Education. *Procedia - Social and Behavioral Sciences*, 103:657–664, 2013. doi:10.1016/j.sbspro.2013.10.385.

22 M Prensky. Digital Game-Based Learning. *McGraw-Hill, New York*, 1, 2001. doi:10.1145/950566.950567.

23 Mustafa Sırakaya and Didem Alsancak Sırakaya. Trends in Educational Augmented Reality Studies: A Systematic Review. *Malaysian Online Journal of Educational Technology*, 6:60–74, 2018. doi:10.17220/mojet.2018.02.005.

24 UNESCO. *Information and communication technology in education: a curriculum for schools and programme of teacher development*. UNESCO, Paris, 2002.

25 Hsin-Kai Wu, Silvia Wen-Yu Lee, Hsin-Yi Chang, and Jyh-Chong Liang. Current status, opportunities and challenges of augmented reality in education. *Computers & Education*, 62:41–49, 2013.

26 Hsin-Kai Wu, Silvia Wen-Yu Lee, Hsin-Yi Chang, and Jyh-Chong Liang. Current status, opportunities and challenges of augmented reality in education. *Computers & Education*, 62:41–49, 2013.

27 Janchai Yingprayoon. Creative Mathematics Hands-on Activities in the Classroom. In *Proceedings of the 13th International Congress on Mathematical Education*, pages 759–760. Springer, 2017.

28 M Zyda. From visual simulation to virtual reality to games. *Computer*, 38(9):25–32, 2005. doi:10.1109/MC.2005.297.

CodeCubes: Coding with Augmented Reality

Bárbara Cleto
Escola Superior de Tecnologia, Instituto Politécnico do Cávado e do Ave (IPCA), Barcelos, Portugal
a13993@alunos.ipca.pt

Cristina Sylla
Research Centre on Child Studies Universidade do Minho, Braga, Portugal
cristina.sylla@ie.uminho.pt

Luís Ferreira
2Ai, School of Technology, Instituto Politécnico do Cávado e do Ave (IPCA), Barcelos, Portugal
lufer@ipca.pt

João Martinho Moura
Escola Superior de Tecnologia, Instituto Politécnico do Cávado e do Ave (IPCA), Barcelos, Portugal
jmoura@ipca.pt

Abstract

CodeCubes is interface that uses Augmented Reality to stimulate Computational Thinking in young students. The visual programming blocks are replaced by paper cubes that have an Augmented Reality marker on each face. Each marker represents a programming instruction. The game is composed of three levels. It consists of programming a car course in a racetrack, driving from the start to the final goal.

Code Cubes takes advantage of the physicality offered by Augmented Reality technology. We present the design and development of the game, focusing on its main characteristics and describing the various development stages. We also present the first results obtained by exploring Code Cubes. The results were positive, showing the potential of Augmented Reality interfaces in learning scenarios.

2012 ACM Subject Classification Computing methodologies → Mixed / augmented reality

Keywords and phrases Tangible Interfaces, Augmented Reality, Computational Thinking, Games

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.7

Funding This work was partly founded by Portuguese national funds (PIDDAC), through the FCT – Fundação para a Ciência e Tecnologia and FCT/MCTES under the scope of the project UIDB/05549/2020.

Acknowledgements We are thankful to the students who collaborated in the creation and implementation of CodeCubes, as well as the students of the Club of Programming and Robotics for testing Codecubes and for the improvement suggestions that they presented.

1 Introduction

Digital games are increasingly gaining relevance in educational contexts, their educational potential has been acknowledged, and consequently, their use as an educational tool inside the classroom has increased [29]. The same phenomenon is seen when it comes to Augmented Reality (AR), and although it is not a new technology, it is now commonly used within educational contexts, challenging traditional education [7]. The combination of augmented reality interfaces with educational content, offers students new possibilities of interacting with the real and the virtual worlds. The possibility of overlapping virtual elements, generated by a computer, and the real world, allows the learning experience to be less static and the interaction to be made in real-time, increasing the efficiency and the appeal of teaching

and learning [16]. Adding an AR component to the games that are already used within the classroom, will offer students new resources that will allow them to work and cooperate in order to solve problems and come up with their own solutions, narratives, and connections [29].

CodeCubes is a digital game with an AR interface developed to teach scientific principles of computational thinking, through a problem-solving approach. In order to solve the tasks, the player needs to explore, experiment and interact with CodeCubes putting to practice a trial /error method. The use of AR and Virtual Reality (VR) change the process of learning from passive to active, allowing active, real-time interaction with the learning contents [22].

This approach aims at introducing basic programming concepts to children through experimentation. The combination of the AR technology with paper cubes, provides a new and interactive way of exploration, creating a motivating and engaging experience that allows students to learn while playing.

In this article, we present the development process of CodeCubes, detailing its conceptualization and planning. The paper is structured as follows: section 2 presents the theoretical framework that sustains the research, section 3 describes the development of the various prototypes. section 4 presents results obtained in the first tests. Finally, section 5 presents final observations and suggestions for future work.

2 Background and related work

Children and young adults, the so called "digital natives" [22], interact most of the time and (apparently) fluidly with digital media, feeling comfortable in sending messages, playing online, and browsing the internet. However, only a few of them can create their own (digital) content, either games, animations, or simulations. It is as if they can "read" but not "write" [27]. Digital fluency demands not only the ability to talk, browse and interact with digital content but also the capacity to project, create and invent with the new media [27, 26] which implies solving problems, projecting systems, and understanding human behavior [35]. Computational thinking [8] does not only mean knowing how to program, but to think and find solutions for the arising problems using the fundamentals of computer sciences, skills that everyone should acquire, just like reading, writing, and arithmetic [35].

The introductory teaching of computer sciences in schools is changing the existing paradigm and shifting the acquisition of programming language to the acquisition of more generic computational thinking skills [14]. At the same time, there is a greater emphasis on playful approaches that increase the motivation and involvement of the students [14]. Currently, there are many kits, robots and/or toys and digital platforms targeting children and young adults that teach concepts related to logic, algorithms, and programming and that can be divided into three major groups: i) physical or tangible interfaces (in which every component is tangible), ii) digital interfaces (for computers or mobile devices that do not have any physical component) and iii) hybrid interfaces (composed of both virtual and physical components) [36]. The physical/tangible kits can be subdivided into two groups, with or without electronics. The hybrid kits can be subdivided into blocks of tangible programming or blocks of virtual programming [36]. Regarding the digital platforms, there are examples such as LOGO [21], one of the first tools developed to teach the basic concepts of computational thinking; Scratch [30], and SratchJr [31] for younger children (ages 5 to 7 years), two interfaces developed by the Lifelong Kindergarten Group at the MIT Media Lab. Other, such as Blockly (Google developers) [6] or the platform Code.org [11] offer a set of games, whose concept is to make the characters execute a specific task, using visual programming blocks, to teach concepts linked to computational thinking [35]. MIT App Inventor [3], initially created

by Google and maintained by the MIT Media Laboratory, is a tool for mobile computing, which also uses block-based visual programming. There are also games for iOS and Android systems, such as Lightbot [18] (there is a Web version as well). Regarding the hybrid kits or interfaces that use tangible programming blocks, one example is the Coding Awbie Game by OSMO [12]. Other well-known examples of tangible programming environments are Lego® Mind-storms [17] and Lego® WeDo [34], Project Bloks [24] or littleBits [19] but although these environments promote creativity and collaboration, they are generally expensive as they are built, using electronic and mechanical components [28]. There are also activities, such as Computer Science Unplugged [5] for introducing the basic concepts of computational thinking as well as computer science concepts in a playful way, without having to use the computer and instead using "unplugged" activities. Some projects such as AR Spot or AR Scratch [25], T. Maze [33] or AR-Maze [15]add augmented reality to hybrid interfaces. The next section presents the CodeCubes development process.

3 CodeCubes

The development of CodeCubes is aligned with the hybrid approach using tangible programming blocks. In the following we present its development process, describe aspects related to usability and interaction, and present the implementation of the various prototypes.

3.1 First Idea

CodeCubes targets students that are starting to learn programming. It combines (physical) paper cubes with AR technology for teaching basic programming concepts. The users manipulate the CodeCubes AR markers, which are glued on physical programming blocks (tangible interface) that represent the instructions of programming. The development process followed a user centered design methodology, involving a small group of students, aged between 13 and 14 years old that collaborated in both, the creation and in the testing of the application, giving suggestions and feedback. This allowed us to adjust different design aspects and make the necessary changes that resulted in the creation of several prototypes.

The prototypes described here were based on games from the Code.org platform [11] and followed the game mechanics of Code.org – Angry Birds – Classic Maze [1], in which the player uses "drag and drop" blocks of visual programming in order to program and overcome the proposed challenges[9].

In the developed prototypes, visual programming, drag and drop blocks were replaced by 3D paper cubes, which have an AR marker on each face that represents basic instructions for programming. Prototype 1 supports programming six instructions: start, up, right, left, down, and end, whereas prototype 2 supports programming four instructions: left, right, up, and down. In order to execute and visualize the programmed actions, the user clicks on the play button, (prototype 1), and on the SPACE key (prototype 2,) if playing on a computer, or by tapping the screen when using a mobile device.

3.2 Implementation of the first idea

The first implementation was carried out in Scratch [30] and aimed at building a house with geometric figures. The players had to program a previously defined path moving a square form to reach a triangle form located at the end of the path in order to build a house [9].

The movement of the square form was programmed, using drag and drop visual programming blocks. These blocks were later replaced by physical (tangible) blocks that represented the programming instructions. Additionally to the conception and implementation of the

this idea, the students drew the figures (square and triangle) to be codified so that, after their recognition, they can fulfill the expected instructions, which, in this case, are of creating movement.

It is intended that CodeCubes allows its users to create their markers, print them and build the cubes using paper, scissors, and glue, a process that will give them a sense of ownership. Therefore, we are encouraging hands on learning, in which users interact with physical objects instead of being limited to a screen with digital content [13].

The object recognition is made by using the camera of a cellphone/tablet or the webcam of a computer. After tracking the objects, the image is captured and processed, and the instructions are executed. The movement of the square is shown according to the sequence in which the objects were placed, allowing the users to visualize and receive feedback in real time of the programming.

3.3 First prototype

The first prototype of CodeCubes was created and developed on the game engine Unity 2017.4.0f1 (64 bits) [32] and the platform AR Vuforia [4] to the operating systems Windows and Android.

The prototype allows carrying out six instructions: start, end, right, left, down and up. Each instruction associated with a face of the cube; each face as an AR marker which is detected by the camera of a mobile device.

For the markers to be easily detected by the vision system, several textures with different patterns were used and associated in each instruction. Therefore, the app will more acutely detect the different markers in situations that might be more adverse when it comes to luminosity [8].

This differentiation makes it possible for the simultaneous visualization of different markers at the same time. The platform Vuforia also contains a detector of the marker's image features, with a quality scale for detection. The different textures and patterns, applied to the several markers, allow for a higher number of visual characteristics, in a superior number to 90%, in the scale of the qualification of the platform Vuforia [4].

The game consists of programming a sequence of actions, setting them in the correct order. The cube's face that represents the intended instruction is placed in front of the camera. To execute and see the programmed actions, the user presses the play button.[9]

Figure 1 shows the execution of the right instruction. The player starts by placing the cube, with the START side up, placed towards the camera; after the recognition of the AR marker, an AR cube can be seen. By approximating a cube with a new instruction, the AR cube changes color, indicating its recognition. By pressing PLAY, the instruction is executed, and the user sees the virtually simulated programming carried out with the tangible cubes.

Figure 1 Instructions and execution with the app CodeCubes.

In this phase, the tests were focused on usability; the first results of the tests were positive, and they were related to the motivation of the use of technology. The most limitation that was found was related to the use of the app under certain light conditions, which made it

more difficult for the recognition of the markers. It is important to pay attention to the light conditions and to be careful with the surface of the cubes in order to avoid the unwanted reflection that might interfere in the marker's recognition.

3.4 CodeCubes Game

The CodeCubes Game was another prototype created using the NyARToolkit [20] library for Processing [23], an open source library of augmented reality for Java, was used; the library Ani [2] was also used, in order to create the animations and transitions used in the program.

The game developed consists of three levels and in each of them, only the course that the car must go through (Figure 2) changes.

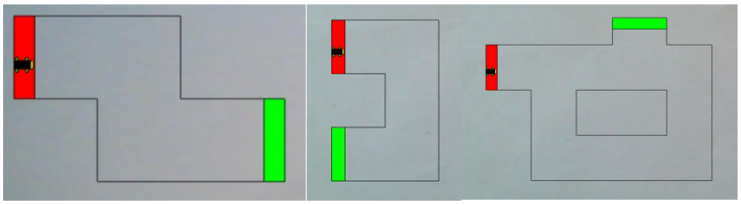


Figure 2 Game levels: Level 1 (left), level 2 (center) and level 3 (right).

The goal is to control the movement of the car, which is over the starting line of a racing track, reaching the end of the road, using an appropriate programming sequence of actions. To move the car, the player must place the AR marker that represents the instruction intended to be carried out in front of the camera [10].

The progress in the game is made by moving the car over the course and reaching the finish line; to do so, the player can execute instructions one after the other or program and execute an instructed sequence.If the player does not execute the instructions or the sequence of correct instructions, but if the car reaches the finish line, it is possible to change levels [10]. This strategy has the goal of motivating students to play while they learn by the process of trial and error and experimenting with different solutions. Therefore, the performance of the students depends solely on the time of execution of each task and allows the student to interact, experiment, and make mistakes without the concern of earning points or losing lives. At any moment, the player can start the level again; the game ends when the three levels are concluded.

The interaction is made by touching the screen, in mobile devices, or with a click of the mouse pad, in the computer version. Four screens have been developed; the starting one, the end one, the gaming screen (in AR mode), and the one to change levels.

In the bottom left corner, there are the buttons to restart the level or to activate and see the execution of the given instructions.

The racing track on the bottom left corner was developed for the pieces to be placed in the upper area. Each time that an object is placed and recognized, the arrow that corresponds to the action appears on the upper left corner. By associating arrows to the objects, the interaction with the game is improved. After objects placement and recognition, instruction processing starts, and the car moves across the track.

The system executes the instructions sequentially, by the physical order they were placed. There is a waiting time between the execution of the given instructions for the user to understand what is being executed and what the user is seeing.

After the sequence execution, and as soon as the car gets to the finish line, a screen appears, one that congratulates the player and that allows the user to play a new level.

One must stress the fact that the level always changes when the car reaches the finish line, whether the programmed sequence is correct or not. The player can restart the level and program each of the instructions one by one or the whole sequence.

4 Exploratory study

The tests were carried out with nine students from the Programming and Robotics Club, RoboESAS – Clube dos Pequenitos, aged between nine and thirteen years old. The average age was ten years old, four boys and five girls, and they are currently in the second year of the programming and robotics club.

With these tests, one intended to detect eventual difficulties of interaction with the game and the interest, motivation, and impact that the use of technology of Augmented Reality (AR) may have in learning.

The activities proposed to students were to perform the same challenge (level 1 of the game CodeCubes). That consists of programming the path of a car on a track so that it can reach the goal, using the Scratch platform and programming one of sever-al robots used in the club. Also, students used the Code.Org platform to play Classic Maze - Angry and explored the developed AR game – CodeCubes. [10]

The instruments and technics for gathering data were: i) direct observation and photo/audio records of the tasks that the participants had to carry out; and ii) a questionnaire and interviews with the participants. The questionnaire was divided into four parts: 1) characterization of the participants, 2) opinion of the participants about the app that they tested, 3) evaluate the impact that AR technology may have in the motivation for learning, and 4) evaluate and classify the activity that they preferred to carry out. All those who participated in the sessions responded to the questionnaires [10].

The results show that, although the participants did not have contact with this type of technology and did not know what it was, they did not show any difficulties in the interaction with the AR game, nor in playing with the AR markers intuitively and autonomously, as pieces of programming [10].

The participants were interested and curious; they were not resistant to participating in the study, nor in using CodeCubes, which turned out to be intuitive and easy to use, without being necessary to explain how it should be used previously.

The activity of programming with AR was the one that the participants enjoyed the most, side by side with programming robots. It was also verified that, on the first level, the students initially preferred to execute the code one instruction after the other; in the following levels, they would set several objects, keeping in mind the course that the car had to go through while manipulating the objects [10]

The enthusiasm of the participants was visible, and it was clear that they were receptive to the use of this type of technology within the space of the classroom, through books, and that it would motivate them in learning the content of other subjects.

5 Conclusions and Future Work

In this article, we described the development process of CodeCubes, that uses Augmented Reality (AR) and physical blocks for teaching basic programming concepts.

The development of CodeCubes had the primary goal of exploring an educational resource based on AR, on the increase of the potential of Computational Thinking (CT) in children aged between 4 and 14 years old, in formal and informal educational contexts, and thus evaluate if this type of technology can, indeed, improve learning. By exploring a new space for gaming and bringing new paradigms of interaction to the classroom, one hopes to have contributed to a better understanding and utilization of AR technology as an environment for learning.

For future work, it would be interesting to allow users to create their own physical markers and tangible objects to manipulate the content of the virtual game. It would be desirable to implement CodeCubes to AR HMDs (head-mounted display) in order to improve the interaction for the fact that both hands would be free for the paper cubes handling. One possible solution would be to create a version without markers using ARCore or creating a version with Leap Motion, which would allow for a type of interaction made with gestures.

Another possible path would be the application of a multiplayer version in order to expand the interaction between the workgroups by creating a collaborative environment and therefore analyze how augmented reality technology can influence the students' social skills; this multiplayer mode might also promote the collaborative resolution of problems.

The suggestions received from the participants of the study include adding sounds to the games, creating more characters, and adding a scoring system. Another aspect that should be improved is the game's appearance and graphic details, by making the characters (or the car) 3D in AR. Moreover, adding more levels, so that students can learn more and gradually, in a more consolidating way, by inserting a new block of instructions that allows the use of structures of repetition and selection, could also be more productive.

References

1 Angry Birds – Code.org – Classic Maze. `https://studio.code.org/hoc/1`, 2019. [Online; accessed 2019/03/15].

2 Ani library Homepage. `http://www.looksgood.de/libraries/Ani/`, 2019. [Online; accessed 2018/02/15].

3 Appinventor. `https://appinventor.mit.edu/`, 2019. [Online; accessed 2019/06/10].

4 AR Vuforia platform. `https://www.vuforia.com`, 2018. [Online; accessed 2018/09/30].

5 Timothy Bell, Jason Alexander, Isaac Freeman, and Mick Grimley. Computer science unplugged: school students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology*, 13, January 2009.

6 Blockly Homepage. `https://developers.google.com/blockly/`, 2019. [Online; accessed 2019/09/30].

7 M. Bower, C. Howe, N. McCredie, A. Robinson, and D. Grover. Augmented reality in education – cases, places, and potentials. In *2013 IEEE 63rd Annual Conference International Council for Education Media (ICEM)*, pages 1–11, 2013.

8 José Cerqueira, Bárbara Cleto, João Martinho Moura, and Cristina Sylla. Visualizing platonic solids with augmented reality. In *Proceedings of the 17th ACM Conference on Interaction Design and Children*, IDC '18, page 489–492, New York, NY, USA, 2018. Association for Computing Machinery. `doi:10.1145/3202185.3210761`.

9 Barbara Cleto, João Martinho Moura, Luís Ferreira, and Cristina Sylla. Codecubes-playing with cubes and learning to code. In *Interactivity, Game Creation, Design, Learning, and Innovation*, pages 538–543. Springer, 2018.

10 Bárbara Cleto, Cristina Sylla, Luís Ferreira, and João Martinho Moura. "play and learn": Exploring codecubes. In *EAI International Conference on Technology, Innovation, Entrepreneurship and Education*, pages 34–42. Springer, 2019.

11 Code.org: What will you create? https://code.org/learn, 2019. [Online; accessed 2019/06/10].

12 Coding Awbie. https://www.playosmo.com/en/coding/, 2019. [Online; accessed 2019/03/15].

13 Anna Fusté Lleixà. *Hypercubes: learning computational thinking through embodied spatial programming in augmented reality; Hyper cubes; Learning computational thinking through embodied spatial programming in augmented reality.* PhD thesis, MIT, 2018.

14 Anna Gardeli and Spyros Vosinakis. *The Effect of Tangible Augmented Reality Interfaces on Teaching Computational Thinking: A Preliminary Study*, pages 673–684. Springer International Publishing, 2020. doi:10.1007/978-3-030-11932-4_63.

15 Qiao Jin, Danli Wang, Xiaozhou Deng, Nan Zheng, and Steve Chiu. Ar-maze: a tangible programming tool for children based on ar technology. In *Proceedings of the 17th ACM Conference on Interaction Design and Children*, pages 611–616. Association for Computing Machinery, June 2018. doi:10.1145/3202185.3210784.

16 M. Tumerkan Kesim and Yasin Ozarslan. Augmented reality in education: current technologies and the potential for education. *Procedia - Social and Behavioral Sciences*, 47:297 – 302, 2012. Cyprus International Conference on Educational Research (CY-ICER-2012)North Cyprus, US08-10 February, 2012. doi:10.1016/j.sbspro.2012.06.654.

17 Lego Homepage. https://education.lego.com/en-us/support, 2019. [Online; accessed 2019/06/10].

18 Lightbot Homepage. http://lightbot.com/flash.html, 2019. [Online; accessed 2019/03/15].

19 LittleBits Homepage. https://www.littlebits.cc/getting-started, 2019. [Online; accessed 2019/06/10].

20 NyARToolkit library Homepage. https://nyatla.jp/nyartoolkit/wp/?page_id=166, 2019. [Online; accessed 2018/02/15].

21 Seymour Papert. *Mindstorms: Children, Computers, and Powerful Ideas.* Basic Books, Inc., USA, 1980.

22 Marc Prensky. Digital natives, digital immigrants part 1. *On the Horizon*, 9:1–6, September 2001. doi:10.1108/10748120110424816.

23 Processing Homepage. https://processing.org/, 2019. [Online; accessed 2018/02/15].

24 Projectbloks Homepage. https://projectbloks.withgoogle.com/, 2019. [Online; accessed 2019/06/10].

25 Iulian Radu and Blair Macintyre. Augmented-reality scratch: A children's authoring environment for augmented-reality experiences. In *Proceedings of IDC 2009 - The 8th International Conference on Interaction Design and Children*, pages 210–213. Association for Computing Machinery, 2009. doi:10.1145/1551788.1551831.

26 Mitchel Resnick. Sowing the seeds for a more creative society. *Learning and Leading with Technology*, 35, January 2007. doi:10.1145/1518701.2167142.

27 Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. Scratch: Programming for all. *Commun. ACM*, 52(11):60–67, November 2009. doi:10.1145/1592761.1592779.

28 Alpay Sabuncuoğlu, Merve Erkaya, Oğuz Turan Buruk, and Tilbe Göksun. Code notes: Designing a low-cost tangible coding tool for/with children. In *Proceedings of the 17th ACM Conference on Interaction Design and Children*, IDC '18, page 644–649, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3202185.3210791.

29 Karen Schrier. Using augmented reality games to teach 21st century skills. In *ACM SIGGRAPH 2006 Educators Program*, SIGGRAPH '06, page 15–es, New York, NY, USA, 2006. Association for Computing Machinery. doi:10.1145/1179295.1179311.

30 Scratch Online Homepage. https://scratch.mit.edu/, 2019. [Online; accessed 2019/06/10].

31 ScratchJr Online. https://www.scratchjr.org/, 2019. [Online; accessed 2019/06/10].

32 Unity (game engine). https://unity3d.com, 2018. [Online; accessed 2018/09/30].

33 Danli Wang, Cheng Zhang, and Hongan Wang. T-maze: A tangible programming tool for children. In *Proceedings of the 10th International Conference on Interaction Design and Children*, pages 127–135. Association for Computing Machinery, 2011. doi:10.1145/1999030.1999045.

34 WeDo 2.0 – Support – LEGO Education. https://education.lego.com/en-gb/support/wedo-2, 2019. [Online; accessed 2019/06/10].

35 Jeannette Wing. Computational thinking. *Communications of the ACM*, 49:33–35, March 2006. doi:10.1145/1118178.1118215.

36 Junnan Yu and Ricarose Roque. A survey of computational kits for young children. In *Proceedings of the 17th ACM Conference on Interaction Design and Children*, IDC '18, page 289–299, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3202185.3202738.

The Use of ARM-Assembly Language and a Raspberry Pi 1 B+ as a Server to Improve Computer Architecture Skills

Vitor Manuel Ferreira
Instituto Politécnico de Viana do Castelo, Portugal
ferreira@estg.ipvc.pt

Pedro Pinto
Instituto Politécnico de Viana do Castelo, Portugal
INESC TEC, Porto, Portugal
pedropinto@estg.ipvc.pt

Sara Paiva
Instituto Politécnico de Viana do Castelo, Portugal
sara.paiva@estg.ipvc.pt

Maria José Azevedo Brito
Instituto Politécnico de Viana do Castelo, Portugal
Centro de Linguística da Universidade Nova de Lisboa (CLUNL), Portugal
mjazevedo@estg.ipvc.pt

Abstract

Prompting students' interest and engagement in learning environments is crucial to achieve the best results. Academia and educators in general are constantly adapting materials and methodologies in order to maximise the acquisition of contents by their students. In this case-study, a new teaching/learning methodology is presented and evaluated through a final questionnaire survey. This case-study aims to understand students' efficiency and motivation levels regarding a new teaching/learning methodology adopted in the second module of a Computer Systems and Architectures course attended by first-year Computer Sciences undergraduates. The new teaching/learning methodology relies on a specific programming language - ARMv6 assembly - to improve students' efficiency levels, and an innovative always-visible in-class mobile test scenario, implemented through a low-cost computing platform - Raspberry Pi 1 B+ - as a server, mimicking as much as possible a real-life environment, so that students believe they are working on real hardware, thus enhancing their motivation levels. The results of the questionnaire survey allowed to infer that the use of a specific programming language, such as ARMv6 assembly, coupled with a new always-visible in-class mobile test scenario were in fact efficient in raising the levels of motivation among Computer Sciences students and, consequently, improved their skills in Computer Architecture.

2012 ACM Subject Classification Computer systems organization

Keywords and phrases ARM-assembly language, Raspberry Pi, always-visible in-class mobile test scenario, Computer Architecture skills, students' efficiency and motivation levels evaluation

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.8

1 Introduction

According to Dunne (2017) [4], the best way to understand how a Computer works – more specifically, how its Central Processing Unit (CPU) works – is through the use of its assembly language, because it is "... the computer programming language closest to (its) CPU's machine language". Although an assembly language is "... unique to a particular CPU design" and, therefore, "not portable from one CPU manufacturer or model to another", what is interesting to highlight is the author's claim that:

First International Computer Programming Education Conference (ICPEC 2020).
Editors: Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões; Article No. 8; pp. 8:1–8:11

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

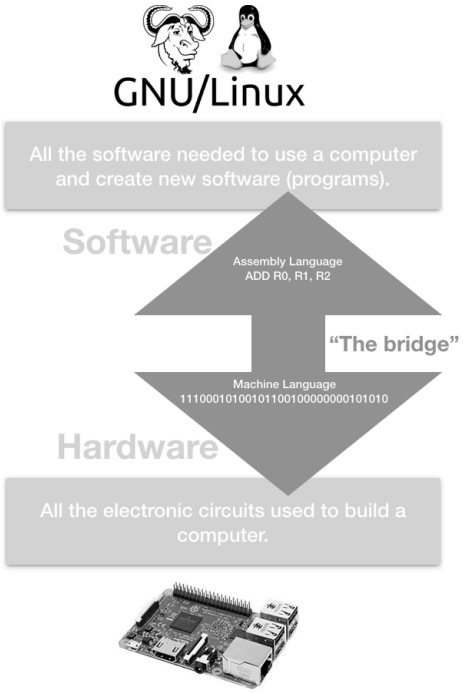


Figure 1 Assembly language, the bridge between hardware and software (adapted from Dunne (2017) [4]).

> "... assembly language is primarily a bridge of understanding between programmers and computer engineers" (Figure 1).

Consequently, since the main goal of the *Computer Systems and Architectures* (CSA) course is to develop skills associated with "... the relation between software and hardware and how programming tasks are executed by hardware" [12], students will be able to learn "... computer architecture and internal processor organization through the writing of assembly programs" [1].

2 Reasons for choosing a Raspberry Pi computing platform

The choice for a low-cost Raspberry Pi computing platform, with a RISC architecture - ARM processor, was made based on Clements's (2010) [3] and Dunne's (2017) [4] claim. In fact, we could have used an emulator [7] instead of real hardware, but "... students do not want to use hypothetical hardware, because they feel it is unrealistic and does not give a true picture of the real world they will soon be entering" [3]. On the other hand, the same authors also argue that "... ARM architecture is an excellent vehicle for teaching computer architecture" [3] because it is "... one of the most popular CPUs currently in production" [4] and used worldwide; the truth is that we all carry a smartphone with a RISC processor in our pocket [8]. Moreover, according to Dunne (2017) [4], choosing a Raspberry Pi computing platform brings further advantages, such as:

1. "Professional quality" – Using a RISC ARM CPU computing platform with a GNU/Linux operating system is "... a common work environment for developing real-time embedded systems";

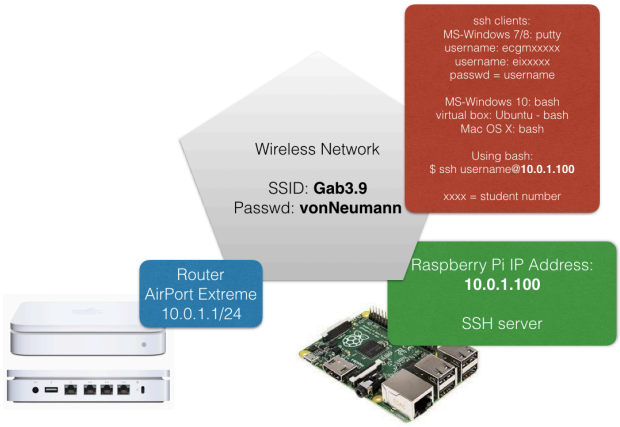


Figure 2 The new in-class scenario.

2. "Edit, Compile, Link and Execute" – Since students typically use Integrated Development Environments (IDE) to develop software, what apparently happens is that they do not realise the key steps involved in software development, namely *Edit*, *Compile*, *Link* and *Execute*. Therefore, in the CSA course, students "... explicitly perform each of these steps separately so (they) can learn the role of each program - editor, assembler, linker" [10] used to create the executable program;
3. "Inexpensiveness" – Most importantly, the low cost of such a platform will allow students to easily acquire it and be able to work outside the classroom environment.

3 The new mobile test scenario

Figure 2 shows the new mobile test scenario used in the classroom environment.

Within this test scenario, a Raspberry Pi 1 B + was permanently used in overclocking at 900 MHz, in order to be as fast as possible without compromising the system stability (Figure 3). To make this test scenario as similar as possible to a real-life environment, a Raspberry Pi was used as ssh server to be able to serve, simultaneously, about 40 students per classroom. With the purpose of keeping the test scenario in students' minds, the portable/mobile test scenario was consistently taken to the classroom instead of having a Raspberry Pi somewhere in the school's wireless network.

However, to manage to serve a practical class of approximately 40 students, it was necessary to use an adequate router. We started with a Linksys WRT54GS Wireless-G broadband router, yet, it was not able to guarantee a persistent connection with the Raspberry Pi, due to the high number of students in class. Therefore, we decided to try out another router – Apple's AirPort Extreme – and, from then on, the test scenario worked without incident.

On the server side, we created an account per student, so that each student could work on the Raspberry Pi using a ssh client at the following command line:

```
$ ssh studentUsername@10.0.1.100
```

With the entire test scenario fully operational, the teaching/learning strategy was grounded on Tanenbaum & Austin's (2013) [11, p. 1] premise:

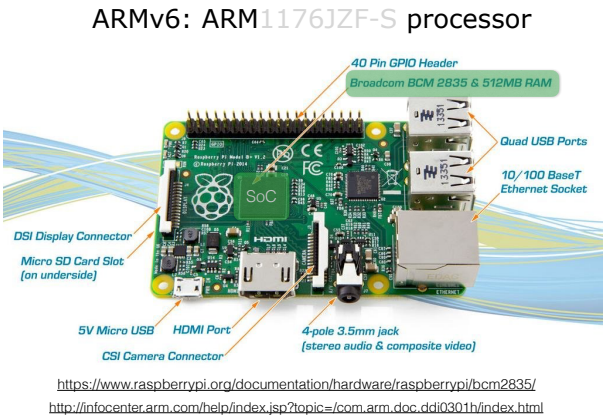


Figure 3 The Rasberry Pi 1 B+.

> "The electronic circuits of each computer can recognize and directly execute a limited set of simple instructions into which all its programs must be converted before they can be executed. These basic instructions are rarely much more complicated than (1) Add two numbers; (2) Copy a piece of data from one part of the computer's memory to another; (3) Check a number to see if it is zero."

Based on this premise, students were challenged to create three simple assembly programs capable of (1) adding two numbers, (2) transferring data from RAM to registers and registers to RAM, and (3) determining whether a number is equal to zero, respectively. At this stage, students were able to learn how to edit the source code of each program, using the vi(m) text editor; how to compile, using the GNU Assembler; how to create the executable program, using the GNU Linker; and how to see the output of each program, using the bash "Exit-Status" variable. All the commands used in each cycle are shown in Figure 4.

Noticeably, the aim of the CSA course is not to make students experts in assembly programming, but rather to allow them to learn computer architecture by writing small programs in assembly, as also claimed by Ibanez (2013) [5]:

> "The idea is not to become a(n Assembly programmer) master but understand some of the details of what happens underneath."

Therefore, students were instructed to follow the first five online tutorials from the above author [5, 6], covering Tanenbaum & Austin's (2013) [11, p. 1] premise.

Finally, to actually see each of these basic instructions in action, all the assembly programs were run, step-by-step, through the GNU Debugger [9].

4 Perception of students' efficiency and motivation levels

In order to perceive and understand students' efficiency and motivation levels regarding the adopted teaching/learning methodology, a questionnaire survey was used to collect data related to:

1. Level of effort;
2. Contribution to learning process;
3. Instructor's skill and responsiveness;
4. Course content.

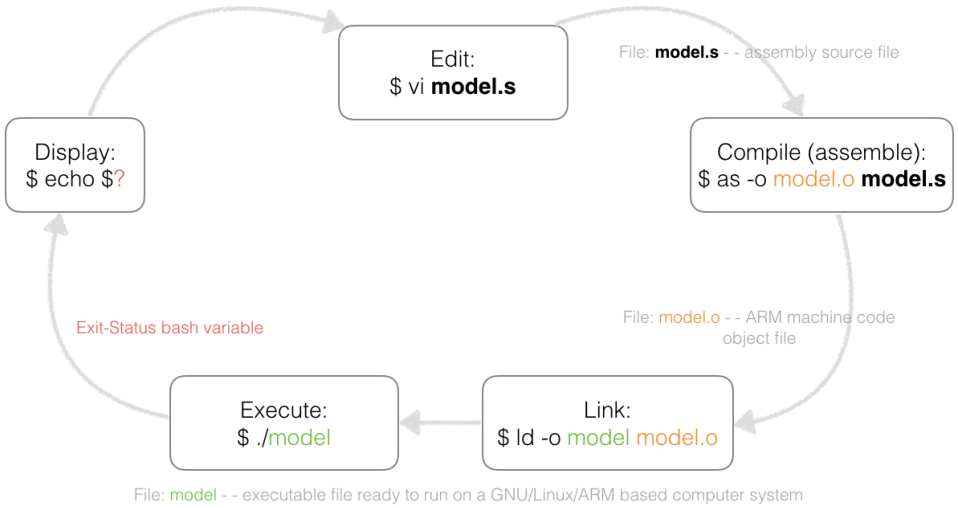


Figure 4 Workflow to test each assembly program into the bash GNU/Linux Operating System.

Since the first two items measure Satisfaction, we used the following scale ratings: Poor, Fair, Satisfactory, Very good and Excellent. As for the last two items, which measure Attitudes and Opinions, our choice was the Likert scale [2]: Strongly disagree, Disagree, Neutral, Agree or Strongly agree.

At the end of the questionnaire survey, two simple open-ended questions were added. The first question focused on the most useful or valuable aspects of the course module, whereas the second one elicited suggestions on how to improve the course module.

Furthermore, to find out how many students actually bought a Raspberry Pi to work outside the classroom environment, a final Yes/No question was asked.

In addition to the questionnaire survey, for purposes of comparison and attestation of the results obtained as regards students' efficiency levels, we also took into consideration the Academic Registry data included in the official CSA course final reports. These reports reflect a final assessment (FA) based on both practical and theoretical approaches. On the practical side (P), students had to submit a technical-scientific report for each practical assignment; on the theoretical side (T), students had to take a written test. The formula for this final assessment is as follows: $FA = 0.5 * P + 0.5 * T$.

5 Sample selection

The students who participated in this study were attending two Computer Sciences undergraduate degrees at the Polytechnic Institute of Viana do Castelo, namely Computer Engineering (CE) and Computer Graphics and Multimedia Engineering (CGME) (Figures 5 and 6).

In the academic year of 2018/2019, the total number of students from these two undergraduate degrees evaluated in the CSA course was precisely 100 (67 CE students and 33 CGME students). However, as a previously established inclusion criterion, students would

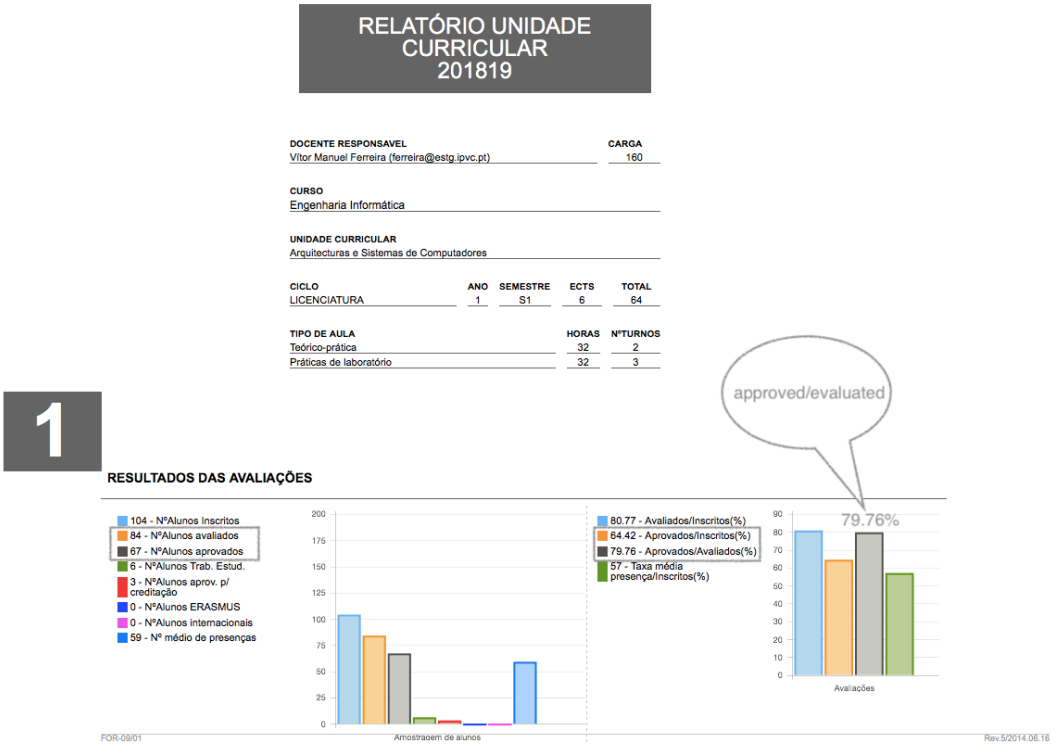


Figure 5 2018/2019 CSA final report concerning the Undergraduate degree in Computer Engineering (CE) – Data retrieved from the official Academic Registry (no English version available).

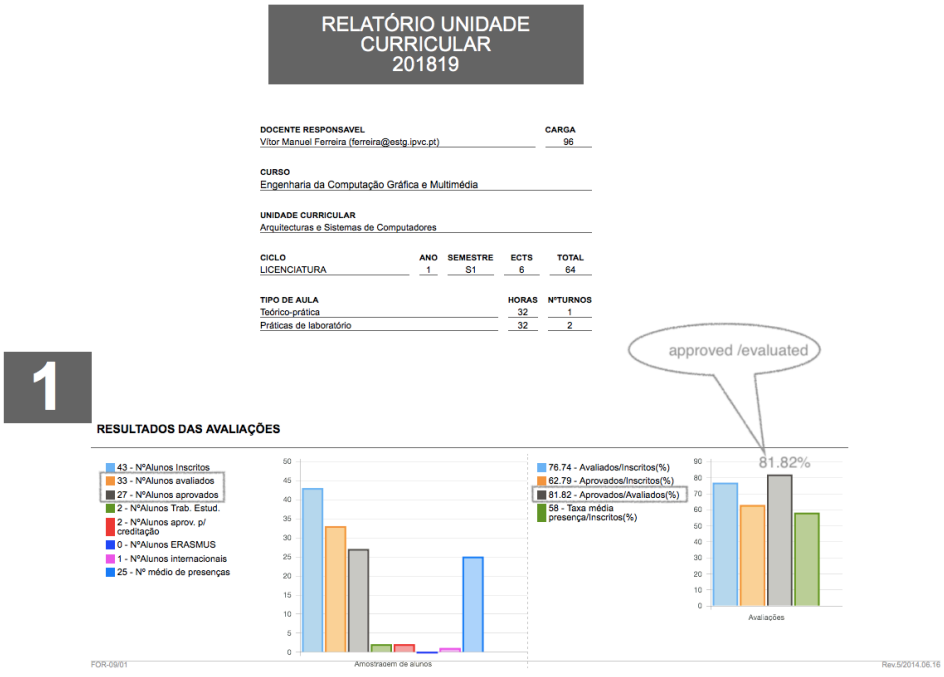


Figure 6 2018/2019 CSA final report concerning the Undergraduate degree in Computer Graphics and Multimedia Engineering (CGME) – Data retrieved from the official Academic Registry (no English version available).

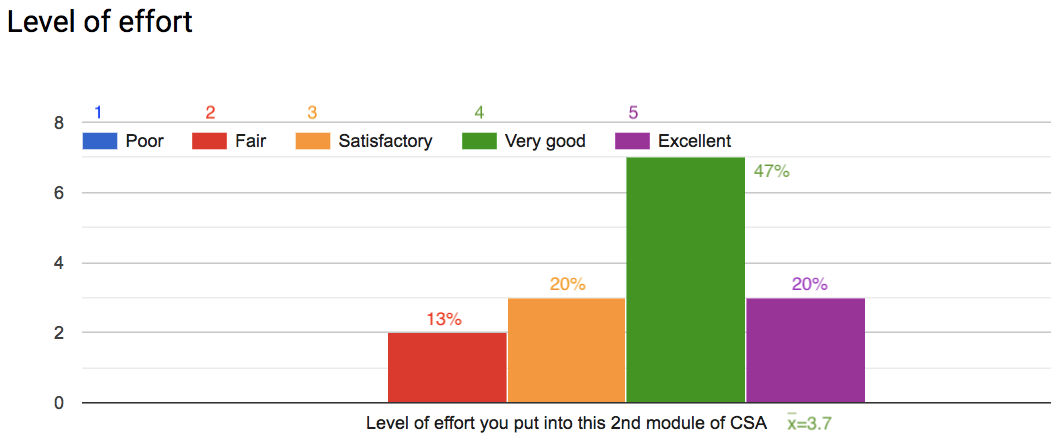


Figure 7 Self-reported level of effort invested in this 2nd module of CSA (15 responses).

need to have achieved a minimum final grade of 13 in a 0–20 scale in order to be eligible to participate in the study. Therefore, only 44 students were invited to participate, representing 44% of all the assessed students. Among these 44 participants, we obtained 15 valid answers, accounting for 15% of all the students evaluated in the course.

6 Analysis and discussion of the data collected

As both CSA final reports (Figures 5 and 6) demonstrate, overall, 80% of the evaluated students were approved. Although indirectly, this fact allows us to infer that the level of students' efficiency is in line with the students' self-reported level of effort (Figure 7) and the replies given as regards the contribution of the course to their learning process (Figure 8).

The data collected in the second item of the questionnaire survey, filled in by the 15-student sample with the main goal of retrieving evidence of the contribution of the CSA course to the students' learning process, are summarised in Figure 8 and provide evidence of: (1) *Level of skills/knowledge at start/end of the course* - although students referred that the level of knowledge at the beginning of the course was satisfying ($MEAN = 2.6$), at the end they rated it as very good ($MEAN = 4.1$); (2) *Level of skills/knowledge required to complete the course* and (3) *Contribution of the course to students' skills/knowledge* - although most students confessed that the course is demanding ($MEAN = 3.7$), most students also agreed that the final level of knowledge acquired was a result of having attended the course ($MEAN = 4.2$).

The data collected in the third item of the questionnaire survey, filled in by the 15-student sample with the main goal of revealing evidence of the instructor's skill and responsiveness, are summarised in Figure 9 and provide evidence of: (1) *Instructor's Effectiveness* - most students agreed (47%) and strongly agreed (33%) that the instructor was an effective lecturer/demonstrator; (2) *Quality of online tutorials used* - most students agreed (53%) and strongly agreed (33%) that the tutorials used were clear and well organized; (3) *Level of motivation and interest provided by the new testing scenario with Raspberry Pi* - most students agreed (53%) and strongly agreed (37%) that using a Raspberry Pi as a server, simulating a real-life environment, stimulated their interest; (4) *Useful feedback and prompt grades output* - most students agreed (53%) and strongly agreed (27%) that grading was prompt and that they had useful feedback.

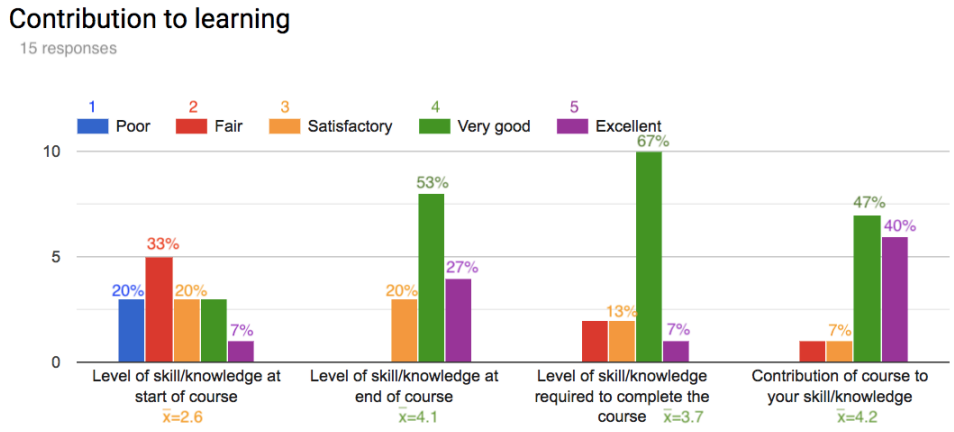


Figure 8 Contribution to the learning process (15 responses).

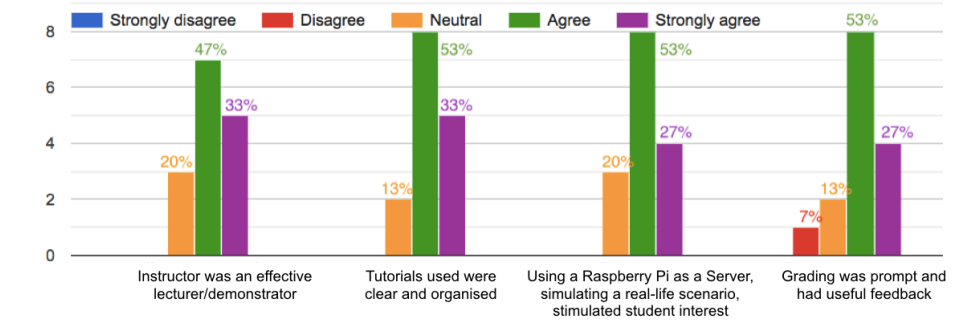


Figure 9 Instructor's skill and responsiveness (15 responses).

The data collected in the fourth item of the questionnaire survey, filled in by the 15-student sample with the main purpose of collecting evidence of Course Content, are summarised in Figure 10 and offer evidence of: (1) *Clarity of the final skills to be achieved* - most students agreed (60%) and strongly agreed (33%) that the learning skills were clear; (2) *Course content organization and planning* - most students agreed (27%) and strongly agreed (53%) that the course contents were well planned and organized; (3) *Course workload* - most students agreed (57%) and strongly agreed (20%) that the course workload was appropriate; (4) *Level of students' participation according to the course organization* - most students agreed (60%) and strongly agreed (27%) that the course was organized in a way that allowed all students to fully participate.

For the two open-ended questions (with non-mandatory answer) - (1) *What aspects of this course module were most useful or valuable?* and (2) *How would you improve this course module?* - summarised in Figure 11, we only obtained 2 answers for the first question (1) and 1 answer for the second (2): (1) "*Understanding how a computer really works, and learning a bit of low level programming*" and "*The use of tutorials for students' self-learning*" were, in the participants' opinion, the most useful and valuable aspects of this course; (2) this course module can be improved with "*More exploration of assembly programming*".

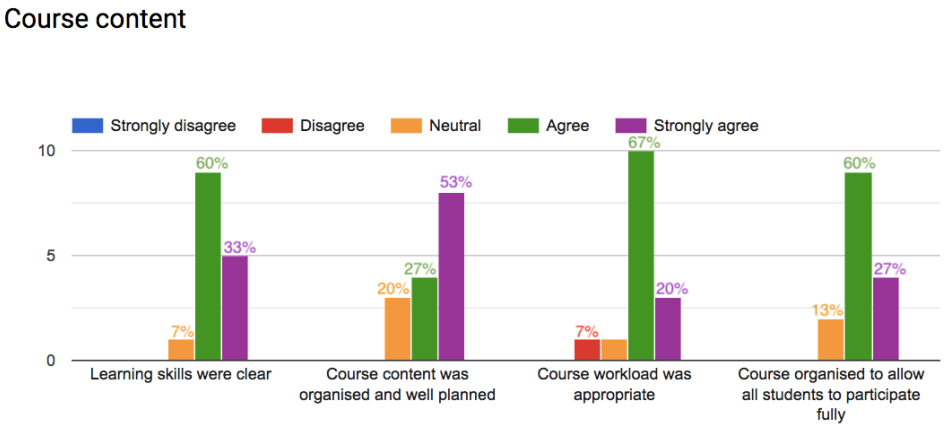


Figure 10 Course content (15 responses).

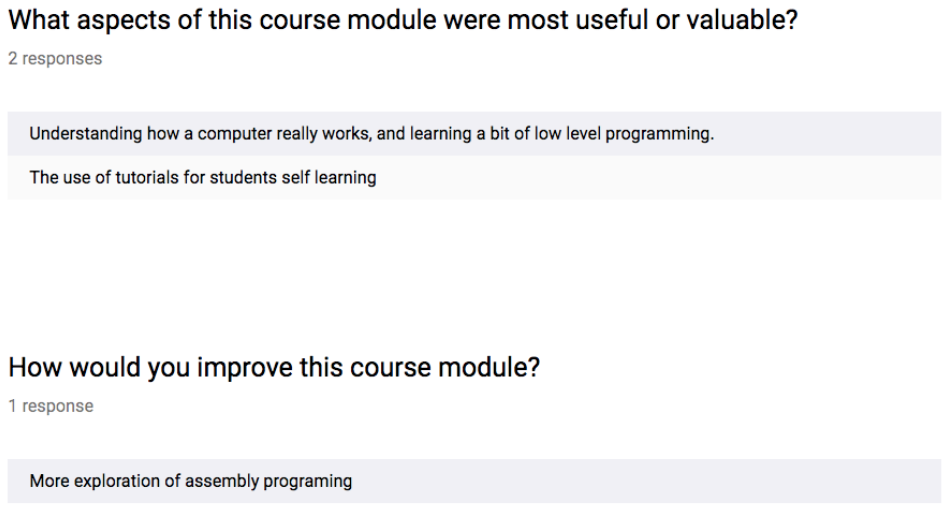


Figure 11 Two optional open-ended questions.

Regarding the last question of the questionnaire survey, filled in by the 15-student sample with the main goal of bringing to light evidence whether the low-cost of the Raspberry Pi platform allowed them to buy it so that they could work outside the classroom environment, data are summarised in Figure 12 and demonstrate that more than half of the students (53%) bought a Raspberry Pi to work from home.

7 Conclusions

Despite the relevance of the results obtained in this study and discussed in the previous section, it is important to underline that we only obtained feedback from 34% of the 44 students invited to answer the survey. Therefore, in order to consolidate the positive results, it would be justifiable to reinforce the previous invitation to participate in the questionnaire survey, in order to obtain the answers from the remaining non-responding students and, thus, prove that this sample is indeed adequate to draw valid conclusions. In any case, we are confident and very convicted that the results obtained in this study clearly show that

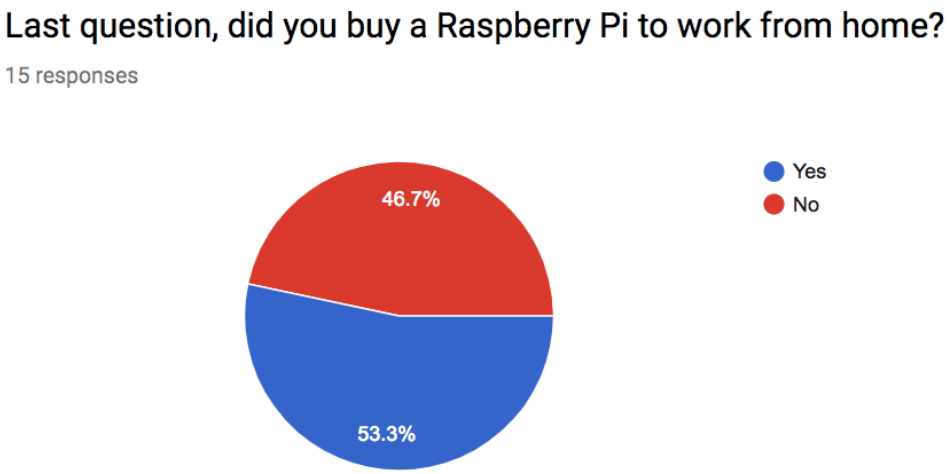


Figure 12 Last question, regarding the purchase of a Raspberry Pi to work from home (15 responses).

the teaching/learning strategy carried out in the second module of the CSA course was in fact efficient in raising the levels of motivation and interest among our Computer Sciences students, and that the use of a test scenario always present and visible in the classroom and based on a low-cost platform, such as Raspberry Pi, made all the difference. If any doubt persists, an approval rate of 80% of all the effectively evaluated students is certainly a very good indicator of the success of the teaching/learning strategy adopted in this second module of *Computer Systems and Architectures*, attended by first-year undergraduate students at the Polytechnic Institute of Viana do Castelo.

References

1 Patricio Bulić, Veselko Guštin, Damjan Šonc, and Andrej Štrancar. An FPGA-based integrated environment for computer architecture. *Computer Applications in Engineering Education*, 21(1):26–35, 2013. doi:10.1002/cae.20448.

2 Seung Youn (Yonnie) Chyung, Katherine Roberts, Ieva Swanson, and Andrea Hankinson. Evidence-based survey design: The use of a midpoint on the likert scale. *Performance Improvement*, 56(10):15–23, 2017.

3 A. Clements. ARMs for the poor: Selecting a processor for teaching computer architecture. In *2010 IEEE Frontiers in Education Conference (FIE)*, pages T3E–1–T3E–6, October 2010. doi:10.1109/FIE.2010.5673541.

4 Robert Dunne. *Assembly Language Using the Raspberry Pi: A Hardware Software Bridge*. Gaul Communications, 2017.

5 R. F. Ibáñez. ARM assembler in Raspberry Pi - Chapter 1,2,3,4 and 5, 2013. URL: https://thinkingeek.com/2013/01/09/arm-assembler-raspberry-pi-chapter-1/.

6 R. F. Ibáñez and William J. Pervin. *RASPBERRY PI ASSEMBLER*. Online, 2017. URL: http://tiny.cc/v20m7y.

7 G. Malhotra, N. Atri, and S. R. Sarangi. emuARM: A tool for teaching the ARM assembly language. In *2013 Second International Conference on E-Learning and E-Technologies in Education (ICEEE)*, pages 115–120, September 2013. doi:10.1109/ICeLeTE.2013.6644358.

8 Muhammad Ali Mazidi, Sarmad Naimi, Sepehr Naimi, and Shujen Chen. *ARM Assembly Language Programming & Architecture (Volume 1)*. MicroDigitalEd.com, 2013.

9 MicroDigitalEd. ARM Assembly Programming Using Raspberry Pi GUI, 2017. URL: `https://bit.ly/2JrkJaK`.

10 Robert G. Plantz. *Introduction to Computer Organization: ARM Assembly Language Using the Raspberry Pi*. Online, 2018. URL: `http://bob.cs.sonoma.edu/IntroCompOrg-RPi/intro-co-rpi.html`.

11 Andrew S. Tanenbaum and Todd Austin. *Structured Computer Organization*. Pearson Prentice-Hall, New Jersey 07458, 6 edition, 2013. URL: `https://goo.gl/N2YQc3`.

12 Hamid S. Timorabadi. Reduced complexity processor for teaching computer architecture. In *Proceedings of the Canadian Engineering Education Association (CEEA) Conference June 3-6, 2018 Vancouver BC*, 2018.

Turing – Inter School Programming Contest: Pedagogical Innovation in Programming Teaching for Middle Schools

Rui Figueiredo
Agrupamento de Escolas Alcaides de Faria, Barcelos, Portugal
ruifigueiredo@aeaf.edu.pt

Bárbara Cleto
Escola Secundária Henrique Medina, Esposende, Portugal
550bcleto@eshm.edu.pt

José Manuel Cerqueira
Agrupamento de Escolas de Barcelos, Portugal
jose.cerqueira@aebarcelos.pt

Abstract

Turing is an interscholastic tournament that aims at promoting the teaching of informatics, particularly the learning of programming through gamification. It is a competition between secondary schools, organized by teachers of informatics, for their own students. Turing was developed due to the lack of tournaments and competitions organized by teachers in this level of education. By contrast, universities and polytechnic institutes regularly organize programming tournaments, aimed at students of both secondary schools and universities. Given that its Turing is a pilot project, the first edition of the tournament will take place in March 2020 and it will occur simultaneously in three secondary schools. The students who are (voluntarily) enrolled in Turing will have an hour and thirty minutes to solve a set of problems in C programming language via Web, through the E-Learning platform Moodle while using the plugin CodeRunner.

2012 ACM Subject Classification Computing methodologies → Logic programming and answer set programming

Keywords and phrases CodeRunner, Contest, Game, Gamification, Programming, Turing

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.9

1 Introduction

Nowadays, computer technology, computational thinking and programming languages are as important as other essential subjects in schools [15]. The relevance of these domains, particularly computational thinking is stated in PISA 2021 Mathematics Framework [1], which refers to

> "The increasing and evolving role of computers and computing tools in both day-to-day life and in mathematical literacy problem solving contexts is reflected in the recognition in the PISA 2021 framework that students should possess and be able to demonstrate computational thinking skills as they apply to mathematics as part of their problem-solving practice. Computational thinking skills include pattern recognition, designing and using abstraction, pattern decomposition, determining which (if any) computing tools could be employed in analysing or solving a problem, and defining algorithms as part of a detailed solution".

[1] http://www.oecd.org/pisa/pisaproducts/pisa-2021-mathematics-framework-draft.pdf

Skills such as analysis, synthesis and evaluation are required and promoted while approaching these subjects. Algorithms and programmes interlink. Programming languages, redundancies of specifications/algorithms, allow the representation through encoding (creating programmes) of the (re)solution of problems. Therefore, when it comes to teaching skills that are related to programming languages, it is important to consider that programming is much more than the writing of a set of code lines in a given language [5] and that to teach to programme is essentially to teach how to think [3].

2 Background

The goal of teaching programming languages is to allow the students to develop skills such as observation, comprehension, analysis, reflection, logical thinking and autonomy, consequently acquiring basic tools and elementary knowledge that are necessary to develop programmes that are able to solve real problems [5]. However, the students reveal some difficulties in syntactic, conceptual and strategic knowledge [11]. These difficulties are related to several factors, namely the unfamiliarity of syntax, natural language, mathematical knowledge and imprecise mental models [11]. Furthermore, they reveal some difficulty in structuring the problems that are proposed to them, even when these are similar to decision-making situations that constitute the normal scenario of their daily routines [5, 14]. Thus, this problem is posed to every single intervenient of the teaching-learning process; teachers ask themselves what to teach and how to do it [14] and the students wonder how to achieve the necessary skills, given that programming implies thinking, solving, defining and formalizing [14]. It is also important that the students are motivated and involved in the process of learning how to programme, given that this area requires a considerable amount of effort in the first stages of learning [8]. The complexity and difficulty of the process results, at times, in demotivation and in the quitting from the programming courses by some students [16]. This factor is a matter of concern for those who are committed to teach subjects such as Introduction to Computer Programming, Programming Languages and other similar subjects. There are several strategies to address the problems that were identified [16, 12, 18, 5, 14], which encompass several types of computer systems that support the learning of programming, recurring to visual representations and animations of algorithms, programming languages based in icons, systems of intelligent tutors, and microworlds of learning [5], games, gamification, programming tournaments, among others.

2.1 Gamification

A new approach to learning [10] named Game Based Learning (GBL), looks at enhancing the motivation and the involvement of students in matters related to learning through educational games. These games, whose strengths are fantasy, curiosity, challenges and control are called Serious Games (SG) [20]. A more recent trend considers that students can also create their own games and, by doing so, develop solutions to problems and therefore, other skills in the areas of Computational Thinking and/or programming. As the research in this area is developed, new concepts are created, such as Edutainment, “Games for Change” [19], Playful Learning or Learning through play [13] e Gamification [4], in which elements of game design are used and included in contexts that are not necessarily related to the game. These concepts, particularly Gamification, were used as the starting point for the creation of a programming tournament which, along with the automatic evaluation of programming problems, would appeal to the students and motivate them when it comes to this particular

area of study. Several studies have demonstrated that Gamification, when used in appropriate conditions, can create a favorable environment to the students and potentially increase the interest of the students in programming [9].

2.2 Supporting tools for the learning of programming (with Learning Management System – LMS)

Programming implies a way of studying which is considerably different from other subject areas, by demanding intensive practicing and a solid background of mathematical knowledge (such as number theory, logic, and others) and problem-solving. Attending classes and studying with books is not enough; programming demands intensive work done outside the classroom [5]. The use of an LMS system that might be applied in any other place, beyond the curricular timetable as part of autonomous studying can potentiate the learning of programming [5]. The positive reinforcement in the answers, the fast interaction, the (appealing) Web fact, the 'awards' as a reward of excellent work, and the ability to work anywhere, anytime, are all advantages in the use of a LMS system as a complement of an introductory subject. An LMS system appropriately adjusted and equipped presents the following strengths when it comes to learning and teaching programming:

- Orienting the correct solving of problems, forcing the student to an intensive practice, in which s/he has to follow every stage that encompasses comprehending, characterizing, representing, solving and reflection on the obtained solution [14];
- Allowing the students to create and simulate their own algorithms and the analysis of the results, while also correcting certain aspects that might have been less successfully developed [5];
- Verifying if the algorithms written by the students are behaving correctly with tests designated by the teacher (entry data and expected results), in an autonomous manner and without the concerns of being evaluated [5];
- Allowing the student to self-evaluate his/her knowledge though the simulation and the testing of their resolutions, giving the student an elevated degree of confidence in the system, as well as in his/her own abilities [5].

Finally, another aspect which is more global from a pedagogical point of view is the great advantage of giving prompt feedback, as well as feedback adjusted in time (formative evaluation).

2.3 Automatic evaluation of programming problems

The relevance of the evaluation of the code of a programme written by a student is related to the supply of data to the teachers, which allows them to assess the level of development of the student and therefore guarantee that the learning outcomes are achieved in order to correct courses [17], i.e., to guide teachers in making decisions that direct and determine the educational processes that are developed with their students. To give students the possibility to solve programming exercises by themselves is essential to develop their capacities. In order to fulfil this, the students need feedback, in an immediate and continuous manner, on their progress, as well as on their difficulties. However, providing individual and timely feedback is highly demanding for a teacher [12]. A likely solution for this problem, one that has been effective and a motivational factor, encompasses the use of on-line systems that enable the automatic evaluation of programming problems [18].

3 Related work

There are several initiatives that aim at promoting digital literacy in computer sciences, some more directed at computational thinking, others specifically related to programming activities, in which the programming contests are included. When it comes to computational thinking, endeavors such as Hour of Code[2] and Code Week[3] are normally destined to the general public and allow any person, anywhere, to organize or take part in coding activities. Hour of Code integrates the event "Education Week in Computer Sciences", which takes place in December for students around the world. This event consists in promoting one-hour long programming activities in a worldwide scale. In Europe, Code Week organizes the European Week of Programming every October. It is funded by the European Commission and it aims at taking programming and digital literacy to everyone in an engaging and playful way. Furthermore, Bebras[4]is an international initiative that aims at promoting Informatics, particularly Computer Sciences and Computational Thinking among students of all ages. The first edition of Bebras in Portugal was hosted by the Department of Computer Sciences of the Faculty of Sciences of the University of Porto (Departamento de Ciência de Computadores – DCC/FCUP, Faculdade de Ciências, Universidade do Porto) Regarding contests that are specifically dedicated to programming, the ones that already exist are promoted and held by universities or polytechnic institutes and are exclusively designed to university students, such as: i) Student Tournament of Multilanguages of Aveiro (TECLA[5] – Torneio Estudantil de Computação multiLinguagem de Aveiro), promoted and developed by the Águeda School of Technology and Management (Escola Superior de Tecnologia e Gestão – ESTGA) of the University of Aveiro; ii) Topas[6], a tournament of programming designed for secondary education students, organized by the Department of Computer Sciences of the Faculty of Sciences of the University of Porto or iii) the Informatics Olympics (Olimpíadas da Informática[7]), hosted by the Association for the Promotion and Development of the Information Society (Associação para a Promoção e Desenvolvimento da Sociedade da Informação (APDSI) in collaboration with the Department of Computer Sciences of the Faculty of Sciences of the University of Porto. For the younger public, there is the National Contest of Programming in Scratch: Creating with Scratch (Concurso Nacional de Programação em Scratch: A Criar Com Scratch[8], promoted by the Centre of ICT Skills of the School of Education of the Polytechnic Institute of Setúbal (Centro de Competência TIC, Escola Superior de Educação, Instituto Politécnico de Setúbal), as part of the project EDUSCRATCH, in partnership with the Directorate General for Education (Direção-Geral da Educação – DGE) of the Ministry of Education and Science (Ministério da Educação e Ciência) and the Committee of Protection of Children and Youth (Comissão de Proteção de Crianças e Jovens – CPCJ) of Setúbal. All these tournaments are organized by universities or polytechnic institutes with the purpose of appealing to students for the study of these specific areas. These contests intend to provide the students with an opportunity to show and improve their knowledge and skills in the solving of problems, by looking at programming not just as the activity of writing programmes, but as the act of developing software, implying that it must be perceived as a team activity [6].

[2] https://hourofcode.com/pt/
[3] https://codeweek.eu/
[4] http://bebras.dcc.fc.up.pt/index.html
[5] http://tecla.estga.ua.pt/
[6] https://topas.dcc.fc.up.pt/
[7] http://oni.dcc.fc.up.pt/2019/
[8] http://projectos.ese.ips.pt/eduscratch/index.php/42-videos/526-a-criar-com-scratch-2020

4 Turing – Inter School Programming Contest

Turing is a tournament promoted and organized by a group of teachers that work in several school groups: Agrupamento de Escolas Alcaides de Faria, Agrupamento de Escolas de Barcelos and Escola Secundária Henrique Medina. Every student that attends these schools can individually participate in the tournament. More information can be found `https://turing.pt`. The tournament takes place simultaneously in the headquarters of every school group that the students attend and consists of a test that lasts for an hour and thirty minutes, in which a set of problems is intended to be solved using the C programming language. The online platform that is meant to be used in the tournament combines Moodle[9],vastly used in schools, and CodeRunner[10], a plug-in.

4.1 Moodle

A vast majority of Secondary Schools and Middle Schools has Web servers with Moodle as a support platform for learning and teaching. Moodle is a system for the management of learning. It is a platform with a virtual desktop, in which it is possible to create courses, add resources such as videos, images, documents, databases, forums, among others [8] and that allows, with the version 2.5., to grant medals or badges to the students [8, 1], enabling the teacher to implement Gamification. There are three distinct ways of giving badges: i) they can be given manually (the medals are created and the teacher can choose when and to whom s/he will be giving them to) ii) after finishing a course or iii) after finishing a certain activity as part of a course, with the medal being automatically added to the student's profile. The badges have a name, a description, an image and the name of the person who assigned it [8, 1]. Another technique that helps in the use of Gamification in Moodle is "level up!" [8, 2]. Thereby, by taking advantage of that experience that was already acquired, a model that could integrate Moodle in an easy and clear manner was searched for. The chosen model was CodeRunner, a module that incorporates a question-type plugin, which can be easily integrated in one of the common questionnaires of Moodle, with the advantage of using its inherent abilities of integration and adaptability. Moreover, the positive experience with the use of CodeRunner by the authors of this article when teaching introductory courses of programming was also a determining factor in the choice of the model. The use of Virtual Programming Lab[11] (VPL) was also considered but this model works as an activity of Moodle that includes several features that are considerably complex, such as revision and verifying plagiarism, tools that transcend the needs of the introductory courses of computer programming in both Secondary Education and 3rd Cycle levels (7th to 9th grade).

4.2 CodeRunner

CodeRunner is an open-source module, free for Moodle and developed in the University of Canterbury in New Zealand, which allows teachers to define questions to include in the questionnaires of Moodle, whose answers are encoded in a particular programming language, such as C, C++, C sharp, Java, JavaScript (NodeJS), Python, PHP, Octave (Matlab), among others. On the other hand, the students develop and test their code using a normal environment of development. When they consider that their answer is correct, they submit

[9] `https://moodle.org/?lang=pt`
[10] `https://coderunner.org.nz/`
[11] `https://vpl.dis.ulpgc.es/`

the code, using a Web browser, through the editor that is provided by CodeRunner, which is associated with the given question. A fundamental pre-condition that lies behind CodeRunner is that the questionnaire in which the questions are inserted is executed in the adaptable mode of Moodle, giving immediate feedback to the students on the accuracy of their answers and allowing them to re-send a corrected answer, with the downside of suffering a certain penalty [7].

4.3 Turning Tournament

Turing, by being organized by a group of teachers, is intended to not only encourage the interest for programming within a younger public, but also to be a method of advertising and promoting innovative pedagogical methodologies, as well as good practices in teaching subjects such as Introduction to Programming (and other similar ones) in Secondary Education and in the 3rd Cycle Schools, by showing that it is possible, with the limited resources that schools own to establish ground-breaking systems which complement the processes of teaching and learning that are up-to-date and motivational for students. Turning is also intended to be distinct from other contests organized by Higher Education institutions, by answering directly to the national demands that are part of the Exit Profile of Students Leaving Compulsory Education (Perfil dos Alunos à Saída da Escolaridade Obrigatória), particularly to the Essential Outcomes (knowledge, abilities and attitudes) regarding skills in Computer Programming.

5 Conclusion and Future Work

In this article, several important aspects related to the learning and teaching of programming were discussed. Taking into account the relevance that programming languages have been gaining in schools and societies, a tournament of computer programming, organized by secondary education teachers and targeting students of that same level of education, with some amount of knowledge of programming was also presented in this paper. The tournament named Turing is still at its early stages of creation and it will only take place in March 2020. Given the results that may be achieved in its first edition and after evaluating its impact, some additional research will be carried out regarding the technical (used tools) and pedagogical (strategies and methods involved in learning) levels of the project. In the future, new schools may be included within the same framework in which Turing was developed. There is also the hypothesis of including an exclusive edition of Turing adapted to 3rd Cycle (7th to 9th grade) students. By adding new plug-ins, such as level-up, it will also be possible to use it as a method of evaluating the students who enrol in the tournament.

References

1 Marcelo Claro. Badges no Moodle - Gamification - Moodle Livre, 2014. URL: `https://www.moodlelivre.com.br/tutoriais-e-dicas/potal/tutoriais-e-dicas-moodle/badges-no-moodle-gamification`.

2 Marcelo Claro. Configurar Condicionais no Moodle - Moodle Livre, 2016. URL: `https://www.moodlelivre.com.br/tutoriais-e-dicas/1606-configurar-condicionaisno-moodle`.

3 Ole-Johan Dahl, Edsger Wybe Dijkstra, and Charles Antony Richard Hoare. *Structured programming*. Academic Press Ltd., 1972.

4 Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: defining" gamification". In *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*, pages 9–15, 2011.

5 Anabela Gomes, Joana Henriques, and António Mendes. Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. *Educação, Formação & Tecnologias-ISSN 1646-933X*, 1(1):93–103, 2008.

6 Pedro Guerreiro. A mesma velha questão: como ensinar programação? *Mexico City: UNAM*, 1986.

7 Richard Lobb and Jenny Harlow. Coderunner: A tool for assessing computer programming skills. *ACM Inroads*, 7(1):47–51, 2016.

8 Jorge Adolfo David Castro Monteiro. Ludificação do ensino da programação: Um caso de estudo, 2017.

9 Stamatios Papadakis and Michail Kalogiannakis. Using gamification for supporting an introductory programming course. the case of classcraft in a secondary education classroom. In *Interactivity, Game Creation, Design, Learning, and Innovation*, pages 366–375. Springer, 2017.

10 Marc Prensky. Digital game-based learning. *Computers in Entertainment (CIE)*, 1(1):21–21, 2003.

11 Yizhou Qian and James Lehman. Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE)*, 18(1):1–24, 2017.

12 Ricardo Queirós and José Paulo Leal. Ensemble: An innovative approach to practice computer programming. In *Innovative Teaching Strategies and New Learning Paradigms in Computer Programming*, pages 173–201. IGI Global, 2015.

13 Mitchel Resnick. Edutainment? no thanks. i prefer playful learning. *Associazione Civita Report on Edutainment*, 14:1–4, 2004.

14 Sónia Rolland Sobral and Pedro Cravo Pimenta. O ensino da programação: exercitar a distância para combate às dificuldades, 2009.

15 Seyyed Meisam Taheri, Minoru Sasaki, Jiangcheng Oliver Chu, and Harrison Thuku Ngetha. A study of teaching problem solving and programming to children by introducing a new programming language. *The international journal of e-learning and educational technologies in the digital media (IJEETDM)*, 2(1):31–36, 2016.

16 Paula Tavares, Elsa Gomes, Pedro Henriques, and Maria João Pereira. Técnicas para aumentar o envolvimento dos alunos na aprendizagem da programação. In *VII Congresso Mundial de Estilos de Aprendizagem, CMEA'2016*, pages 1565–1577. Instituto Politécnico de Bragança, 2016.

17 Zahid Ullah, Adidah Lajis, Mona Jamjoom, Abdulrahman Altalhi, Abdullah Al-Ghamdi, and Farrukh Saleem. The effect of automatic assessment on novice programming: Strengths and limitations of existing systems. *Computer Applications in Engineering Education*, 26(6):2328–2341, 2018.

18 Elena Verdú, Luisa M Regueras, María J Verdú, José P Leal, Juan P de Castro, and Ricardo Queirós. A distributed system for learning programming on-line. *Computers & Education*, 58(1):1–10, 2012.

19 Nelson Zagalo and Dionisia Laranjeiro. Brinquedos e jogos digitais para o jardim de infância. *4. o Encontro sobre Jogos e Mobile Learning*, 2018.

20 Michael Zyda. From visual simulation to virtual reality to games. *Computer*, 38(9):25–32, 2005.

Cybersecurity Games for Secure Programming Education in the Industry: Gameplay Analysis

Tiago Gasiba
Siemens AG, München, Germany
Universität der Bundeswehr München, Germany
tiago.gasiba@siemens.com

Ulrike Lechner
Universität der Bundeswehr München, Germany
ulrike.lechner@unibw.de

Filip Rezabek
Siemens AG, München, Germany
filip.rezabek@siemens.com

Maria Pinto-Albuquerque
Instituto Universitário de Lisboa (Iscte), ISTAR, Portugal
maria.albuquerque@iscte-iul.pt

Abstract

To minimize the possibility of introducing vulnerabilities in source code, software developers may attend security awareness and secure coding training. From the various approaches of how to raise awareness and adherence to coding standards, one promising novel approach is Cybersecurity Challenges. However, in an industrial setting, time is a precious resource, and, therefore, one needs to understand how to optimize the gaming experience of Cybersecurity Challenges and the effect of this game on secure coding skills. This work identifies the time spent solving challenges of different categories, analyzes gaming strategies in terms of a slow and fast team profile, and relates these profiles to the game success. First results indicate that the slow strategy is more successful than the fast approach. The authors also analyze the possible implications in the design and the training of secure coding in an industrial setting by means of Cybersecurity Challenges. This work concludes with a brief overview of its limitations and next steps in the study.

2012 ACM Subject Classification Security and privacy → Software security engineering; Security and privacy → Web application security; Applied computing → Interactive learning environments; Applied computing → E-learning

Keywords and phrases education, training, secure coding, industry, cybersecurity, capture-the-flag, game analysis, cybersecurity challenge

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.10

Acknowledgements We would like to thank the anonymous reviewers for the valuable comments and careful reviews. We would also like to thank all game participants as well as our colleagues Jorge Cuellar, Holger Dreger and Thomas Diefenbach for many fruitful discussions.

1 Introduction

Recent years have not only seen an increase of software vulnerabilities leading to cyber-attacks, but also an increase of literature dedicated to the topic. Anyone with interest in attacking a system can just pick up a book or download some tool (e.g., Kali Linux) and start doing potentially destructive actions.

For this reason, software written in the industry, in particular, software for critical infrastructures, must be well designed and developed from a security point-of-view. Such software needs to meet security standards and comply with security coding guidelines. Efforts

in industry aimed to achieve secure software include training, threat and risk modeling, static and dynamic code analysis, and penetration testing. Organizations define their security coding guidelines and adopt secure software development life-cycles with or without tool support. Significant players, e.g., Siemens, initiated the Charter of Trust to establish industry-wide standards in software development. As awareness for the topic increases in the industry, it is –from a Cybersecurity point-of-view– desirable that software developers adopt secure coding practices. In the implementation of secure coding practices, the human factor is the crucial point. This paper addresses the challenge to facilitate the learning of techniques of secure coding and to foster problem-solving skills in conceptualization, design and development of secure software. Our approach to facilitate learning is a serious game: the Cybersecurity Challenges (CSC).

CSC is a new serious game that refines the popular Capture-the-Flag (CTF) format. Gasiba et al. have designed and introduced this training method [7]. CSC targets software developers from the industry. A CSC consists of a collection of challenges and solving one challenge results in being awarded a flag. Teams of software professionals compete in playing a CSC. A typical Cybersecurity Challenges event takes one day and comprises a pool of 204 challenges for players to solve. The order of challenges is defined such that all participating teams in a particular event follow the same sequence of challenges. To determine the winner, flags are translated to points, and the team with the highest amount of points wins the event. The CSCs have a dedicated IT-infrastructure. A coach monitors a CSC event and may provide guidance or give hints to ensure a pleasurable gaming experience and, hopefully, a positive impact on secure coding awareness and skills. Topics of challenges include secure-code patterns, typical weaknesses in code, and the problem-solving skills to identify and eliminate them, or use of cryptographic methods. The goals of the CSC are to raise awareness for the need of secure coding guidelines and transfer knowledge about techniques and tools to be used in secure coding. The Cybersecurity Challenges are designed to train software developers in secure coding.

Cybersecurity Challenges, as defined in [7], are 1) characterized by their specific focus on all aspects related to secure coding, 2) on being designed to address the needs of participants coming from industry, e.g., focused challenges, limited time, focus on industry-specific use-cases. CSC design elements refine the design of Capture-the-Flag. The focus distinguishes CSC from typical Capture-the-Flag (CTF) events: CTF games pose complicated security puzzles to teams, and, often, these games take days, and only a few of the participating teams manage to solve all the challenges. Typically, CTF events address security specialists or students and go beyond typical topics or day-to-day business.

This paper presents the first analysis of data from Cybersecurity Challenges. This study uses data of 9 CSC events that took place exclusively in industrial context between 2017 and 2019. During these events, data from dashboard interaction of a total of 134 CSC participants have been gathered and analyzed. The objective is to understand the interaction that takes place in Cybersecurity Challenges, to identify implications for the game design and the optimization of the gaming experience of participants. Our analysis lays a first step towards measuring the increase in awareness and secure coding skills based on dashboard interactions. Furthermore, we identify player profiles based on these interactions, which give a possible approach for game coaches to direct their help towards individual players or teams. Time is a precious resource in the industry, and thus, the time spent to solve a single Challenge is the point of departure for our analysis.

2 Related Work

This work aims to determine how Cybersecurity Challenges, which are a form of serious games, are played by participants from the industry. A serious game is "a game which is designed with a primary goal that is not pure entertainment" [6]. Serious games have recently gained much attention in the research community as a means to raise information security awareness [7, 14, 15]. For a structured review on information security awareness and the related concepts, see Hänsch et al. [9].

Capture the Flag (CTF) is a serious game genre in the domain of Cybersecurity. Common goals of such serious games are: training purposes, identifying the best students and assessing new employee skills, e.g. potential pen-testers. In a CTF, a series of challenges need to be solved by single players or often teams of players. Various authors have analyzed that CTFs are fun activities with educational value [5, 11]. Game activities may lead to experience in concentration, interest, and enjoyment resulting from increased levels of affective, behavioral, and cognitive involvement with the task at hand [3, 8, 10].

Evaluation of participant performance in a Serious Game is a vital part of the assessment of the game itself [12]. It is critical to understand how to refine and improve the game, but also to know how effective a particular game is to raise secure coding awareness among its participants.

Gasiba et al. [7] discuss the requirements needed to address software developers in the industry as the primary target group. One requirement pertains to the design and planning of the challenges themselves, specifically the time it takes to solve them (challenge solve-time). The rationale is that the challenges presented to the participants should be able to be solved in the amount of time that the event is designed to last.

Mäses et al. [13] propose additional metrics to measure and track participants' progress. In their work, they mostly look at weighted scoring and the time required to solve challenges. These metrics can be used by game designers to increase the effectiveness and efficiency of the learning experience.

Recent work by Andreolini [1] proposes a scoring algorithm based on modeling of trainee activities during cyber range games. A comparison of both trainee scores and desired activities path is the basis for the scoring algorithm. Path activities modeled as directed graphs are analyzed to identify player profiles.

3 Approach and Research Design

Figure 1 depicts the architecture, based on [7], that we have conceptualized, designed, and deployed to implement the CSC game. It comprises of a wireless access point which connects the computers of the players, that run a local virtual machine, to a local server and (optionally) connects to the internet. The server runs a dashboard [4], countdown website and hosts the challenges. The players' local virtual machine also host local challenges. These challenges can be accessed after the game is finished.

In the beginning of the game, the participants are briefed on the game play, are encouraged to build teams (with maximum number of 4 persons), virtual machines are distributed and configured. At the end of the game the winning team is announced, players are given participation certificates an the winning team receives a small coin gift. Additionally, feedback is gathered from the participants on the experience and a short discussion on difficult challenges is held.

The dashboard contains the list of the challenges to be solved by the participants, along with their categories and points. Further challenges are unlocked by solving some previous challenge, e.g. questions on secure coding guidelines to avoid SQL injection are asked after

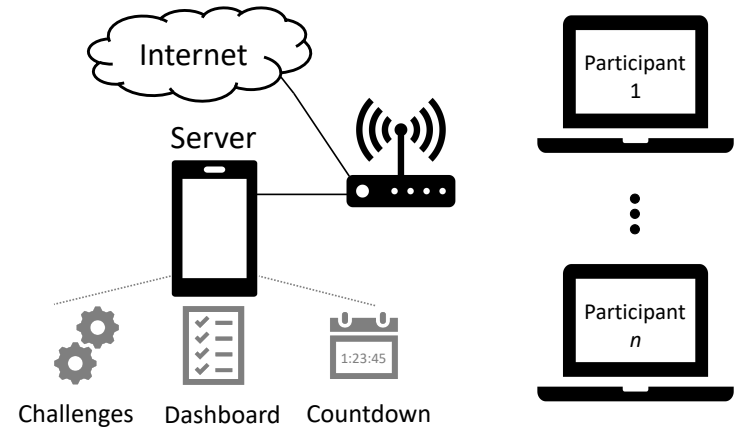


Figure 1 Architecture of Cybersecurity Challenges.

solving the "SQL injection" challenge. All player interactions with the dashboard are kept in a separate log file. Some challenges provide also hints (which cost points) that can be requested by the players. The hints are hosted in the dashboard and their request is also logged.

From 2017 to 2019, the authors have collected data of 9 different CSC events (that took place in four different countries), whereby 134 software developers, pen-testers, and test engineers with ages ranging from 25 to 60 and with an industrial background have participated. Table 1 summarizes the 9 game events in chronological order with the number of participants and the focus domain.

Table 1 Overview of CSC events.

Event No	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
When	2017 Nov	2018 May	2018 Jul	2018 Jul	2018 Sep	2019 Aug	2019 Sep	2019 Sep	2019 Oct
No Players	11	12	6	30	16	14	15	7	23
Focus	Mixed	Web	Web	Mixed	Web	Mixed	Mixed	Web	C/C++

CSCs are tailored to the participants' level of experience, and, with the focus domain to their typical secure coding problems. This tailoring to a focus domain ensures the relevance of the game for the day-to-day work of the participating software professionals. We distinguish three focus domains:

- *Web*: secure coding problems in Web-development with back-end and front-end,
- *C/C++*: secure coding problems in the C and C++ programming language,
- *Mixed*: both web and C/C++ secure coding problems.

CSC challenges belong to one of the six categories:

- *C/C++*: challenges related with C/C++ secure coding guidelines,
- *Comics*: challenges related to general user behaviour presented in a comic style (cf. also [2]),
- *Forensics*: challenges with analytic methods, e.g., the analysis of PCAP files and the traffic captured in these files with tools as, e.g., Wireshark,
- *Python*: secure coding topics specific to Python programming language, i.e., secure coding problems in data analysis,

- *Questions*: topics related to company IT security processes, software life-cycle or specific to secure coding guidelines
- *Web*: questions related to secure coding of Web-applications (both front-end and back-end)

The questions in the CSC game are multiple-choice questions. These include company-internal questions specific to secure coding and internal supporting organization, and also on software code analysis. Possible answers to the questions are multiple choice and the number of tries is limited in order to avoid brute forcing answers.

Also note that over time the collection of CSC challenges has been continuously developed and, at the end of 2019, comprises of a pool of 204 different challenges on the categories detailed above.

Event 1 validated the core CSC design and tested the gaming infrastructure. Over time, also the challenges changed: the Comic challenges were presented in events 2 and 3 and were not part of any further CSC. Note that in the $4th$ event, 3 pen-testers and 2 quality engineers participated beside software engineers. As such, this event had a total of 16% of non-software developers and 84% software developers.

During a CSC event, participants give their (written) consent that data from the game may be used anonymously for scientific purposes. Participants also receive a briefing on how to use the platform and on the game and the game logic. On-site, coaches, are accessible during the whole gameplay to answer questions regarding the setup, usage of tools and to help with the challenges themselves. After the actual game, participants are asked to fill out a questionnaire on the gaming experience and learning outcome.

For this study, only dashboard data is considered. This data was collected during the nine CSC gameplays in the industry. For all of the following dashboard interactions: *correct challenge solve*, *incorrect challenge solve* and *request for hint*, the timestamps were collected from the interactions of the clients with the dashboard and the challenge. For each team, the flags that they captured and the points that the players and teams earned was also collected. All data is anatomized and not traceable back to individual persons.

4 Analysis and Results

In this section presents an initial analysis of the results of the nine CSC games played in an industrial setting as shown in Table 1. The results were pre-processed using Python scripts and then analysed using R-Studio version 1.2.5001. In sub-section 4.1 the challenge solve-time is discussed, sub-section 4.2 focuses on team profiles and in sub-section 4.3 a cross-check between profiles with final score ranking is made. Finally in sub-section 4.4 the limitations of our analysis and threat to validity of our conclusions is considered.

4.1 Time to Solve Challenges

The first analysis is about the solving time for a challenge, i.e., the time spent to solve a challenge. In our setup, it was not possible to collect directly the time each player spends in each challenge. This is an issue that other game data analysts have addressed (for instance, Mäses et al. [13]). Figure 2 visualises our approach to this limitation, which is to measure the solving time for a challenge using dashboard data. The challenge C solve time is computed as the difference $t(C) = t_s - t_x$ in time between the time the challenge was solved t_s with the time of any interaction which was not related to the challenge t_x. We have also added the constraint to the game logic that no dashboard interaction related to the challenge should take place before t_x, in order to guarantee that the player only started working on the challenge after the interaction at t_x.

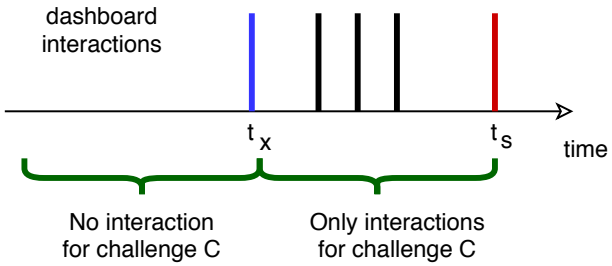


Figure 2 Computing challenge solve time from dashboard interactions.

Table 2 summarizes the challenge solve time for categories of challenges: average (avg.), minimum (min), maximum (max), standard error (stde.), 25% quartil (q25), 50% quartil (q50), 75% quartil (q75), 99% quartil (q99) and kurtosis (k).

Table 2 Detailed Challenge Solve-Time Results.

	avg. (sec)	**min.** (sec)	**max.** (sec)	**stde.**	**q25**	**q50**	**q75**	**q99**	**k**
C/C++	1973	69	6852	201.5	666	1172	2810	6702	3.24
Comics	245	14	1494	41.2	42	105	275	1444	7.22
Forensics	555	10	6772	54.4	81	227	545	4988	19.30
Python	1269	63	6893	176.3	375	743	1844	5553	7.07
Questions	246	3	6904	14.3	23	52	153	3865	40.20
Web	1025	7	6973	65.9	197	492	1173	5876	7.07

The analysis illustrates that different topics and kinds of questions yield different times to respond. The results obtained in this work show that it takes on average about 30 *min* to solve C/C++ challenges, 20 *min* for Python challenges, 15 *min* for web challenges and 4 *min* for multiple-choice questions. This is a clear indicator that C/C++ challenges are harder to solve than web challenges. Generic multiple-choice questions and questions based on comics are less challenging to solve, as expected. Surprisingly, in Table 2, the Forensic challenges, although not the core competency of the players, have been on average solved in only slightly more time than multiple-choice questions. The authors attribute this to the fact that, in order to solve these challenges, specialized tools (in our case, Wireshark) help. Even if participants do not know this tool, they can navigate it and find the appropriate option to solve the problem.

Furthermore, the average time to solve the Comics challenges was observed to be about 4 *min*. It was determined that the participants did not find the comics to be useful during the CSC events (they were even found to be distracting, see [2]). Therefore challenges from the comics category were only present on the 2*nd* and 3*rd* event.

Note that the the two categories Forensics and Questions have high kurtosis values. This is an indicator that there is no given, well known path to the result. The participants might know how to use an appropriate tool, know of a simple method to solve the challenge quickly, or they need to use their own skills to solve it. Potentially also the background and experience of players leads to the differences in time that it takes to solve a challenge. For the Python challenges however, the average time is larger and the kurtosis lower. This indicates that it takes considerable effort to solve such a question, but there is a defined strategy which players may follow to reach the solution in a given time. Similarly for C/C++ and Web categories.

Using these results, a designer of CSC which wants to design an event that lasts 6 *hrs* can use the following guidelines for the agenda: 1 *hr* introduction, 7 C/C++ challenges, 21 questions and 1 *hr* for conclusion. These analysis results also indicate the training levels of participants, their skill sets and the maturity of the topic. Mature tools and knowledge about these tools lead to short solving times. In cybersecurity and secure coding, things change - new tools, new methods, new standards or training efforts have the potential to change the efforts necessary to solve a challenge. From the data observed, the authors think it is necessary to monitor the solving time for the challenges.

4.2 Team Profiles

The second analysis is about team profiles that represent a strategy to deal with the challenges. Here the authors looked at the curves resulting from the normalized cumulative interactions of teams with the dashboard versus the normalized CSC event time (typically 6 hours). Analysis of the team interactions with the dashboard resulted in three team profiles: fast, slow and automated. The last profile (automated) was rejected in our study since it was the results of pentesters (during event no. 2) attacking the dashboard using automated scripts. This was later confirmed by asking the team members directly about the phenomenon in the data. As such, in this work, only two profiles resulting from human interaction and gameplay are considered:

- *Fast* - the interaction takes place mostly at the beginning but wears out as gameplay advances,
- *Slow* - most of the interactions happen towards the end of the gameplay, with fewer interactions at the beginning.

By looking at the resulting curves, the authors have determined that the shape of the curves resembles the following formula:

$$I(\tau) = \frac{a^{\tau} - 1}{a - 1}, \quad a > 0, \quad 0 \leq \tau \leq 1. \tag{1}$$

Note that, τ represents the normalized time and, for $a > 0$ this curve shape formula results in a Slow profile and for $a < 0$ this formula results in a Fast profile. The resulting minimum error average a value of 0.05 and is similar for both Fast and Slow profiles. Although a theoretical explanation for the curve shape formula is not available, it has been shown to produce relatively good results by curve-fitting using a minimum-squared-error algorithm from all the empirical data. Furthermore, this value indicates a sound fit between the model given by equation (1) and the collected data from real CSC events.

Figure 3 shows two examples best-fitting curves, for Fast and Slow profiles, using minimum-squared-error criteria for events 4 and 3. Note that, for reference, the automated profile observed in the CSC event nr 2, although discarded in future analysis, is also part of this figure.

Plotting the normalized team interactions with the dashboard over normalized time (see Figure 3) depicts the two expected team profiles: slow and fast.

This has implications for game design as well as for facilitating and managing the gaming experience by a coach or trainer. Both curves indicate that management of time in the game needs attention and, therefore, a coach should advise and guide participants in case they start fast and have an eye on whether they get stuck in challenges. Participants play a CSC typically only once and timing issues should not deteriorate gaming experience or learning. Thus, a coach or trainer should have a look on the timing aspect. Our analysis also

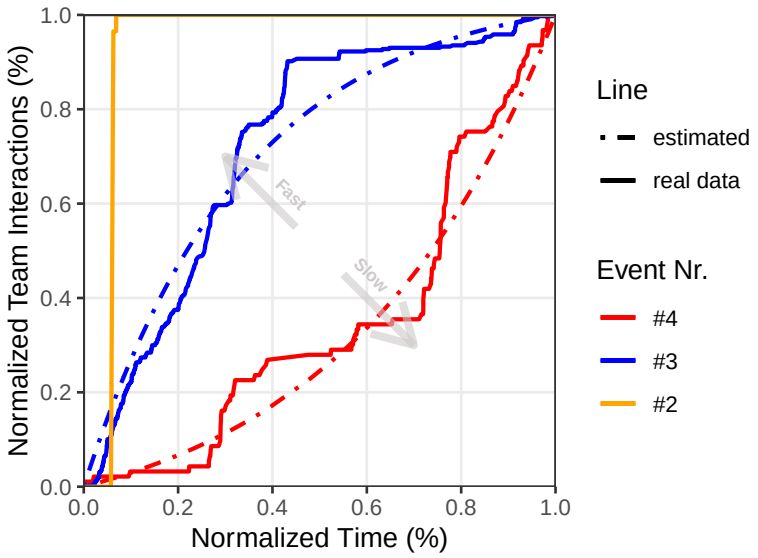


Figure 3 Examples for normalized time vs. normalized total interactions.

prepares a coach for the different strategies individual players have. In further research this might be a topic addressed in game design: it eventually makes sense to provide gamers with more feedback on their timing. Again, with the timing topic, the authors identify another aspect that needs constant monitoring in a CSC and also adequate tool support by the infrastructure.

4.3 Profile and team performance

In this sub-section the gathered data is analysed in terms of the relation between team profile and team performance in the game (in terms of scoring). The authors have thus identified all the corresponding curve types (fast or slow) by means of curve-fitting, for all the teams, and compared them with the final game score ranking. Table 3 summarizes these results; for the nine games played, 4 teams ended in first place with fast profile while 5 teams ended in first place with slow profile, and so on for the remaining places.

Table 3 Ranking of profiles and scores.

Place	1st	2nd	3rd	4th	5th	6th	7th	8th
Fast	4	4	5	4	3	3	2	3
Slow	5	5	4	5	2	2	2	0

The data in Table 3 also shows that, if looking at teams that finish in the first place, 56% belong to the Slow profile and 44% belong to the Fast profile. However, looking at the first, second and third place, the distribution is $48\% - 52\%$ for fast and slow profile respectively. The expected value for the place of Fast and Slow profiles are $E(Fast) = 4.0$ and $E(Slow) = 3.3$, corroborating with the previous observations.

This means that a team having a Slow or Fast profile is not guaranteed to win the game. Nonetheless, the slow profiles do show slightly better results than the fast profiles. Note, as the collection of challenges is individual for each CSC event, the ranking of the teams is used for this comparison of performance and not the number of flags acquired in the challenge.

Further research is needed to establish a relation between number of flags won in a game, number of challenges mastered, profile and learning outcome in terms of awareness, knowledge and skills. Further empirical evidence and analysis is needed for the differences of slow and fast profiles.

4.4 Limitations and Threats to Validity

This study presents a first analysis of data from 9 CSC events which took place between 2017 and 2019 with a total of 134 players. These games have been played on-site in organizations in the business of industrial software engineering. The number of events and the number of participating players is low - as it is inherent in early stages of developing and implementing a novel method as the CSC game. The CSC events, however, have been played as a voluntary training event with software professionals and most of the training events were organized following a request (including remuneration) from the management or a business unit for training of the software professionals. Thus, it can be assumed that the players did their best in solving the challenges, and it can also be assumed that data collected and the analysis results have a high validity. The limitations on the number of games, number of participants, and variations in terms of background and experience are inherent to this kind of industrial setting. We argue, however, that a different context, e.g., in the lab with either students or participants acquired through social media, results would have been different, and validity of the results would be lower.

Further research is needed to clarify the the measurements due to the high standard errors in data. The authors argue, however, that the guiding principles on which data to use and how to collect and analyze these data are promising: data collection and analysis are lightweight, respect the privacy of participants, and allow to monitor the games and the learning outcomes.

The formula for the shape of the interaction curve (Figure 3) also needs a more in-depth analysis and a discussion of the theoretical foundation. Further theoretical analysis is necessary to justify the derived result.

In the present analysis Questions have been considered as one challenge category. However, further investigation is needed in order to separate generic questions and questions that are specific to previous challenges (i.e. that are unlocked by solving some challenge).

Finally, since in every game, individual players were part of a team, the authors have analysed the influence on ranking in terms of team performance and not individual performance. More research is needed, to validate the data on individual performance.

5 Conclusions and Discussion

In this work, Cybersecurity Challenges is presented as a serious game that raises awareness for secure coding, and that trains software developers in secure coding techniques. Using this type of game, first analysis of data from 9 CSC played from 2017 to 2019 in four different countries with software professionals from industry are presented and analysed. Only data from the game dashboard is used in this analysis. The presented results from the analysis are shown to have implications for game design and individualization of Cybersecurity Challenges. The authors present a pragmatic method to analyze Cybersecurity Challenges by using the solve-time for a challenge, i.e. the time it takes to solve a challenge.

The authors identify, based on the analysis of the challenge solve-time the following profiles: automated, slow and fast profile. Automated profiles are discarded in our analysis since they do not reflect human behaviour. Our preliminary results indicate that the slow

profile, with few interactions in the beginning, has advantages over the fast strategy. The method followed to analyze games is pragmatic: it takes only data from the dashboard, no personal data and no linkage to individual persons and to the learning outcome. It is therefore useful for analysing training event played on-side in industry.

Our analysis has implications for the design of Cybersecurity Challenges events. The data on solve-time allows to tailor events to a particular time-frame. E.g., for a 6 *hrs* event, the target should be 7 C/C++ challenges and 21 questions to be solved by the participants. Also, our analysis is useful for the game coaches: monitoring the dashboard allows coaches to provide targeted guidance with the goal to optimize the gaming experience and the learning outcome.

In further research, the authors plan to analyse how the gaming experience contributes to the security levels of software, the levels of secure coding knowledge and of secure coding skills. Additionally, analysis of data from the post-gaming questionnaire will also be conducted in a next step. The argumentation that an awareness measure has implications for level of security in the future is difficult as, e.g. the German Bundesamt für Sicherheit in der Informationstechnik argues in their description of Serious Game for Awareness in the Basic Protection Catalog (Grundschutzkataloge). As such, the next step will also be done to justify that the outcome is worth the effort of playing a Cybersecurity Challenge. Time is a crucial factor in an industrial setting and, therefore, more analysis is needed to be able to optimize the game in terms of gaming experience and increased awareness in secure coding and secure coding skills.

References

1 Mauro Andreolini, Vincenzo Giuseppe Colacino, Michele Colajanni, and Mirco Marchetti. A framework for the evaluation of trainee performance in cyber range exercises. In *Mobile Networks and Applications*, volume 25, pages 236–247, December 2019. doi:10.1007/s11036-019-01442-0.

2 James Barela, Tiago Espinha Gasiba, Santiago Reinhard Suppan, Marc Berges, and Kristian Beckers. When interactive graphic storytelling fails. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, pages 164–169. IEEE, September 2019. URL: https://ieeexplore.ieee.org/xpl/conhome/8932374/proceeding.

3 Paul Cairns. Engagement in Digital Games. In Heather O'Brien and Paul Cairns, editors, *Why Engagement Matters*, pages 81–104. Springer, May 2016. doi:10.1007/978-3-319-27446-1_4.

4 Kevin Chung. CTFd : The Easiest Capture The Flag Framework. URL: https://ctfd.io/.

5 Ian Cullinane, Catherine Huang, Thomas Sharkey, and Shamsi Moussavi. Cyber Security Education Through Gaming Cybersecurity Games Can Be Interactive, Fun, Educational and Engaging. *J. Computing Sciences in Colleges*, 30(6):75–81, June 2015. URL: http://dl.acm.org/citation.cfm?id=2753024.2753042.

6 Ralf Dörner, Stefan Göbel, Wolfgang Effelsberg, and Josef Wiemeyer. *Serious Games: Foundations, Concepts and Practice*. Springer International Publishing, 1 edition, 2016. doi:10.1007/978-3-319-40612-1.

7 Tiago Gasiba, Kristian Beckers, Santiago Suppan, and Filip Rezabek. On the Requirements for Serious Games geared towards Software Developers in the Industry. In Daniela E. Damian, Anna Perini, and Seok-Won Lee, editors, *27th IEEE International Requirements Engineering Conference, RE 2019, Jeju Island, Korea (South), September 23-27, 2019*. IEEE, 2019. URL: https://ieeexplore.ieee.org/xpl/conhome/8910334/proceeding.

8 Schoenau-Fog Henrik. The Player Engagement Process - An Exploration of Continuation Desire in Digital Games. In *DiGRA - Proceedings of the 2011 DiGRA International Conference: Think Design Play*. DiGRA/Utrecht School of the Arts, January 2011. URL: http://www.digra.org/wp-content/uploads/digital-library/11307.06025.pdf.

9 Norman Hänsch and Zinaida Benenson. Specifying IT security awareness. In *25th International Workshop on Database and Expert Systems Applications, Munich, Germany*, pages 326–330, September 2014. doi:10.1109/DEXA.2014.71.

10 Sangkyun Kim, Kibong Song, Barbara Lockee, and John Burton. Engagement and fun. In *Gamification in Learning and Education*, Advances in Game-Based Learning, pages 7–14. Springer International Publishing, 2018. doi:10.1007/978-3-319-47283-6.

11 Kees Leune and Salvatore Petrilli Jr. Using capture-the-flag to enhance the effectiveness of cybersecurity education. In *Proceedings of the 18th Annual Conference on Information Technology Education*, pages 47–52. ACM, 2017.

12 Sten Mäses. Evaluating cybersecurity-related competences through serious games. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*, Koli Calling '19, New York, NY, USA, 2019. Association for Computing Machinery.

13 Sten Mäses, Bil Hallaq, and Olaf Maennel. Obtaining better metrics for complex serious games within virtualised simulation environments. In Maja Pivcec and Josef Gründler, editors, *The 11th European Conference on Game-Based Learning (ECGBL), Graz, Austria*, pages 428–434, October 2017.

14 Andreas Rieb. IT-Sicherheit: Cyberabwehr mit hohem Spaßfaktor. In *kma - Das Gesundheitswirtschaftsmagazin*, volume 23, pages 66–69, July 2018.

15 Andreas Rieb, Tamara Gurschler, and Ulrike Lechner. A gamified approach to explore techniques of neutralization of threat actors in cybercrime. In *GDPR & ePrivacy: APF 2017 - Proceedings of the 5th ENISA Annual Privacy Forum*, Lecture Notes in Computer Science, pages 87–103. Springer Verlag, June 2017.

Ranking Secure Coding Guidelines for Software Developer Awareness Training in the Industry

Tiago Gasiba
Siemens AG, München, Germany
Universität der Bundeswehr München, Germany
tiago.gasiba@siemens.com

Ulrike Lechner
Universität der Bundeswehr München, Germany
ulrike.lechner@unibw.de

Jorge Cuellar
Siemens AG, München, Germany
Universität Passau, Germany
jorge.cuellar@siemens.com

Alae Zouitni
Universität Passau, Germany
zouitni.alae@gmail.com

Abstract

Secure coding guidelines are essential material used to train and raise awareness of software developers on the topic of secure software development. In industrial environments, since developer time is costly, and training and education is part of non-productive hours, it is important to address and stress the most important topics first. In this work, we devise a method, based on publicly available real-world vulnerability databases and secure coding guideline databases, to rank important secure coding guidelines based on defined industry-relevant metrics. The goal is to define priorities for a teaching curriculum on raising cybersecurity awareness of software developers on secure coding guidelines. Furthermore, we do a small comparison study by asking computer science students from university on how they rank the importance of secure coding guidelines and compare the outcome to our results.

2012 ACM Subject Classification Security and privacy → Software security engineering; Security and privacy → Web application security; Applied computing → Interactive learning environments; Applied computing → E-learning

Keywords and phrases education, teaching, training, secure coding, industry, cybersecurity, capture-the-flag, game analysis, game design, cybersecurity challenge

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.11

Acknowledgements We would like to thank the anonymous reviewers for the valuable comments and careful reviews. We would also like to thank all survey participants as well as our colleagues Holger Dreger and Thomas Diefenbach for many fruitful discussions.

1 Introduction

It is widely known that developers (humans) make mistakes during software development which result in bugs [7, 21, 24, 27, 34]. In particular, these bugs can lead to software vulnerabilities that can result in potentially fatal consequences, both for the user of the software, the owner or service provider and the company that sells the software.

Many security standards, e.g. [8, 5, 6, 26, 29, 25], nowadays mandate the implementation of a secure software development life-cycle (e.g [17]), which aims at significantly reducing the number of vulnerabilities in software. In order to be effective, companies should make sure

that their software developers are properly trained in writing secure software, i.e. secure code. Not only is this an important factor in reducing the number of software vulnerabilities, but this is typically checked during audits and certification. As such, specialized training that addresses how to write secure code is needed in order to raise the cybersecurity awareness [13] of software developers in the industry.

Since training of software developers in the industry costs precious time and money, it makes more sense to focus first the effort of training and raising awareness on issues that cause a larger impact to the business [10]. In particular, we are interested in ranking secure coding guidelines for teaching purposes.

In this work we intend to focus on C and C++ programming languages, since they are currently widely used in the industry [28]. For these two programming languages, we use the well known secure coding guidelines (SCG) from Carnegie Mellon University [31] as the basis for the teaching curriculum.

One possible way to prioritize this curriculum, is to base the ranking of the SCG on two steps: 1) the impact rating from real-world software vulnerabilities and 2) the mapping of vulnerabilities to secure coding guidelines. One common and widely used way to rate the impact of software vulnerabilities is to use the common vulnerability scoring system (CVSS) [15]. Online databases, such as the Common Vulnerabilities and Exposure database [23], give extensive lists of known vulnerabilities (CWE - Common Weakness Enumeration), their CVSS scoring and also provides a mapping from these vulnerabilities to secure coding guidelines.

Towards the goal of prioritizing SCG, we need to compute ranking metrics for secure coding guidelines. Note, however, that the goal of this work is not to establish new metrics for secure coding guidelines, but to understand what are the most important SCG that should be taught during awareness training for industrial software developers. However, due to the lack of well-known metrics that combine all the previously mentioned factors, we first define four different CWE metrics, based on well-known mathematical functions (e.g. average value, weighted average, etc.). These metrics are also based on [23], CVSS scoring and also on discussions with cybersecurity experts.

In order to determine and motivate the need for education of secure coding guidelines, we also try to understand the gap between academia and industry. The main question we would like to address here is: are future industry software developers aware which are the secure coding guidelines that have the most relevancy for the industry (i.e. based on real-world data)? This is done by asking last-year university students of computer-science course (not specific to cybersecurity) to rank secure coding guidelines using a likert scale [18] through the means of a questionnaire.

Our main contributions in this work are the following:

- Methodology to compute ranks of secure coding guidelines as basis for prioritization of the education of software developers
- Tables with ranked secure coding guidelines for C and C++ based on real-world data
- Analysis of different ranked SCG, leading to the conclusion that the exact CVSS score values do not significantly contribute to the ranking
- Comparison study between real-world data and student perception of ranking of secure coding guidelines

2 Related Work

In industry several IT security standards, e.g. [8, 5, 6, 26, 29, 25] mandate not only the implementation of secure coding guidelines into the software development life-cycle but also that the developers are properly trained in secure software development. C and C++ are typical and widely used programming languages in the industry [28]. The major secure coding standards in existence for C and C++ are from the Carnegie Mellon University [31] and from MISRA [2, 3].

The importance of secure programming guidelines in the software development life-cycle is discussed by Tabassum et al. and Whitney et al. in [30] and [32] respectively. In order to to raise awareness [13] of software developers on the topic of secure coding guidelines, they study two different methods: ESIDE, an educational IDE plugin, and coaching by security experts. Their preliminary results give indicators that both methods are suitable towards this goal. In a related work, Gadient et al. [9] develop an IDE plugin to detect security code smells as a supportive measure for (Java) software developers. They found out that, out of the 100 applications they investigated, 44 contained security vulnerabilities.

Additionally, many developers lacking knowledge or training in secure coding tend to search online forums on solutions to secure coding problems [33, 24, 20]. It has been shown that the information provided in these online forums can lead to the introduction of further vulnerabilities into the source code [7, 34], if the developers are not aware on how to write secure code. Furthermore, Meng et al. [21] show that the a substantial number of developers does not appear to understand the security implications of coding options and also links this to the lack of cybersecurity training.

Bagnara et al. [4] discuss the MISRA-C guidelines, which are C-specific guidelines composed of safety guidelines and secure coding guidelines. In their work, they distinguish between guidelines that can be verified by automatic tools such as static code analysis, those that require information that is beyond the reach contained in the source code and those that relate to compliance. In [11], Goodall et al propose a method, based on static code analysis, which can be used by software developers to visualize code security. However, in [12] Goseva et al. point out that the coverage of static code analysis tools can vary across different tools. They conclude that one should not rely only on static code analysis tools, otherwise a large number of vulnerabilities can be left undiscovered.

This further points out the need to train software developers in secure coding guidelines, specifically on high impact rules. If software developers are not aware of secure coding guidelines and the issues that can be caused by exploiting vulnerable code, the effectiveness of such kind of tools will be limited. Results presented by Rexxa et al [27] corroborate with these observations. Although focused on web technologies, their results point out that one major reason for software vulnerabilities is the lack of experience and lack of knowledge about secure coding and secure application development.

Recent research results explore promising, new and innovative ways to raise awareness about secure coding to software developers using capture-the-flag methodology [10, 16]. The results presented in this work are directly applicable to this education methodology as a means to prioritize on developed challenges and awarded game points.

3 Outlook of the work

In this section we give a brief overview of our work. It was done in two phases: Phase 1: ranking of SCG using online databases and Phase 2: raking of SCG through questionnaires administered to academia students. Note, this process was repeated for the C programming language and for the C++ programming language.

3.1 Phase 1: ranking of SCG using online databases

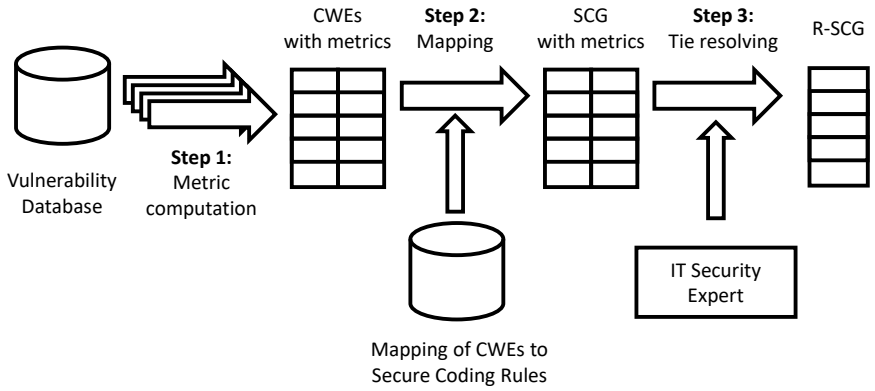


Figure 1 Derivation of Ranked Secure Coding Guidelines.

Figure 1 shows the process that followed in order to derive ranked secure coding guidelines (R-SCG). It consists of the following three steps:

Step 1 compute CWE metric $m^{(x)}(c)$ for each of the four defined metrics (see section 3.3)

Step 2 compute SCG metric based on CWE $\rightarrow$ SCG mapping and then filtering the top 15 SCG by computed metric

Step 3 generate R-SCG table by resolving ambiguous ranks (i.e. using expert opinion for SCG that have the same metric value)

3.1.1 Details on step 1: computing CWE metric

Based on the CVE details online database [23], we have extracted and grouped the CWEs and their corresponding CVSS scores $s(c, \lambda)$. Here c represents the CWE ID and λ represents the observation index, which ranges from 1 to $n(c)$, i.e. the total existing entries (observations) in the database that have CWE with ID= c. At the time that we consulted the online database (May 2019), it consisted of 114.686 observations from 1st January 1999 until 5th May 2019 containing 112 unique CWE identifiers. The computation of the four metric functions $m^{(1)}(c)$ to $m^{(4)}(c)$ will be detailed in section 3.3.

3.1.2 Details on step 2: compute SCG metric based on CWE metric

The MITRE CWE database [22], contains pointers from CWE to the affected SCG from Carnegie Mellon CERT-SEI Secure Coding Guideline database [31]. The mapping provided by this database was used as the mapping rule. Note that, in this database, one CWE can map more than one SCG (see Figure 2). The final SCG metric was taken as the sum of the related CWE metrics multiplied by the CERT-SEI priority level (see sub-section 3.4).

3.1.3 Details on step 3: generate R-SCG table

After step 2, some SCG still had the same computed metric. At this stage, it was decided to disambiguate the tied SCG by gathering input from three different IT cybersecurity experts from the industry. The experts were asked to rank the relative importance of only those guidelines that had the same metric. After this, a table containing the ordered SCG was produced, which we call the ranked secure coding guidelines (R-SCG).

3.2 Phase 2: ranking of SCG by academia students

In order to understand how students in the academia perceive the importance of secure coding guidelines, we have conducted an online questionnaire using Google forms. The number of participants in this questionnaire, which lasted one month and was done in *September 2019*, was 34. The age of the participants ranged from 23 to 30, they were all Master students in computer science (not specializing in cybersecurity), in their second year (last year). All of the participants were familiar with programming in C, and half of them (17 participants) were familiar with programming in C++.

Participants were asked to rank every secure coding rule, which was the outcome of phase 1, in a five point likert scale [18] ranging from "not important" to "very important". For every SCG, the individual likert points were averaged. The resulting SCG were sorted based on the average likert points, resulting in a ranked secure coding guidelines P_C and P_{C++} from academia.

3.3 Metrics

Four different metrics as defined below were used to rate the importance of the CWE in relation to each other (see Table 1). In this table, c represents a CWE ID, $n(c)$ the number of occurrences of incidents in the online database related to c and $s(c, \lambda)$ represents the CVSS score (with values in the range 0..10) present in the online database where c is the attached CWE ID and λ a running index of the entries (in the range $1..n(c)$ entries). A CVSS score is a quantitative severity ranking measure, with 0 being the lowest and 10 being the highest. The reason why four metrics were used, was due to the lack of previous work that gives a metric on SCG based on CVSS scores. Note that all formulas use standard well-know formulas adjusted by the number of occurrences $n(c)$, in order to penalize CWEs that occur more often.

In our work, we define these four metrics as a mean to aggregate the individual CVSS scores into a high-level individual CWE score, i.e. $m^{(x)}(c)$, with x being the selected metric according to Table 1. This metric is then used, as shown in the next sections, to make a further breakdown to individual secure coding guidelines, as shown in Figure 1, step 1.

Table 1 CWE Metrics.

Metric	Description	Formula
#1	Average CVSS Scoring	$m^{(1)}(c) = n(c) \times \frac{\sum_{\lambda=1}^{n(c)} s(c,\lambda)}{n(c)}$
#2	Weighted average CVSS Scoring	$m^{(2)}(c) = n(c) \times \frac{\sum_{\lambda=1}^{n(c)} s^2(c,\lambda)}{\sum_{\lambda}^{n(c)} s(c,\lambda)}$
#3	Worst-case Score	$m^{(3)}(c) = n(c) \times \max_\lambda s(c,\lambda)$
#4	Number of occurrences	$m^{(4)}(c) = n(c)$

3.4 Mapping CWE metrics to SCG metrics

The CWE metrics, as obtained above, are mapped to SCG metrics, as shown in step 2 of Figure 1. In order to achieve this, we used the existing mapping from CWE IDs to SCGs as given by MITRE [22]. It was observed that using this mapping, a single CWE ID can relate to several SCGs. Therefore, we aggregate the final metric computation as given in the exemplary Figure 2. In this example, SCG_1 is referenced by two CWE IDs: CWE_1 and

CWE_3, where each CWE has its own attached metric, as given section 3.3. The resulting SCG metric is then given by $p(SCG_1) \times (m^{(x)}(CWE_1) + m^{(x)}(CWE_3))$, where x in the range $[1, 4]$, and $p(SCG_1)$ represents the SCG priority given by Carnegie Mellon SCG in [31].

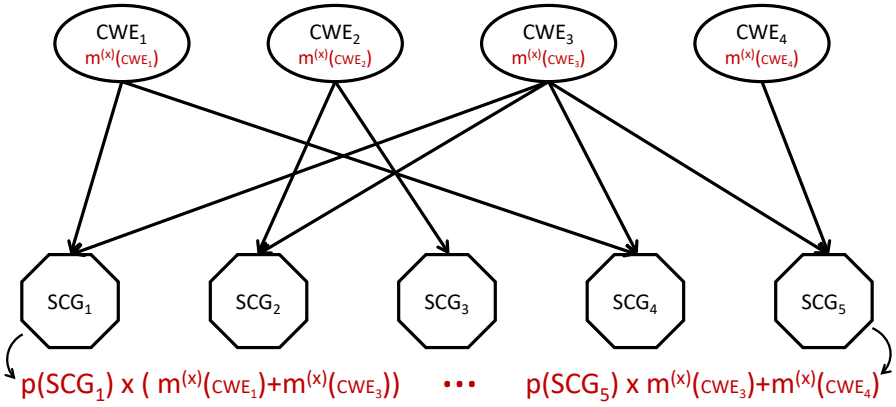


Figure 2 Details on Mapping CWE metrics to SCG metrics.

4 Results

After step 2 and step 3 of phase 1, the results for the C and C++ ranked secure coding guidelines, can be seen in Table 2 and Table 3. Note that in this section, we present the final results, corresponding to step 3 in Figure 1, which are the ranks of the SCG (e.g $R_C^{(1)}$, $R_C^{(2)}$, $R_C^{(3)}$,$R_C^{(4)}$), after computing the different metrics 1..4 for each secure coding guideline. We also present the SCG ranked by the students (P_C) by means of the survey. In the tables, lower numbers indicate higher ranks and higher numbers indicate lower ranks.

Table 2 Top 16 C Ranked Secure Coding Guidelines.

C SCG	$R_C^{(1)}$	$R_C^{(2)}$	$R_C^{(3)}$	$R_C^{(4)}$	P_C
STR38	1	1	1	1	5
EXP34	4	2	2	2	11
STR31	2	3	3	3	2
ARR38	3	4	4	4	6
EXP33	9	5	5	6	12
FIO30	7	6	7	5	1
STR32	8	7	6	7	3
ARR30	12	8	8	10	4
FIO34	10	9	9	8	13
FIO37	11	10	10	9	15
ARR32	13	11	11	11	10
ARR39	14	12	12	12	9
FIO45	16	13	13	13	14
MEM30	5	14	14	14	8
MEM34	6	15	15	15	16
MEM35	15	16	16	16	7

Table 3 Top 15 C++ Ranked Secure Coding Guidelines.

C++SCG	$R_{C++}^{(1)}$	$R_{C++}^{(2)}$	$R_{C++}^{(3)}$	$R_{C++}^{(4)}$	P_{C++}
MEM50	1	1	1	1	5
MEM51	2	2	2	2	7
MEM52	3	3	3	3	1
MEM53	4	4	4	4	3
MEM54	5	5	5	5	4
MEM55	6	6	6	6	15
MEM56	7	7	7	7	10
STR50	8	13	13	13	2
STR51	9	14	14	14	9
EXP53	10	8	8	8	12
EXP60	11	9	9	9	14
EXP54	12	10	10	10	6
EXP61	13	11	11	11	11
EXP62	14	12	12	12	8
STR52	15	15	15	15	13

In order to compare the rankings between themselves, we have computed the Kendall's tau distance metric [19] between the different ranked lists. The Kendall's tau distance is equal to number of exchanges that a bubble sort algorithm needs to apply to one list so that it becomes equal to the other list, i.e. it results in the same ordering of items. A Kendall tau distance of 0 means that the lists contain the elements in the same order. For two lists of size N that are not in the same order, the Kendall tau distance is a value larger than zero and smaller or equal to $N \times (N-1)/2$, i.e. the maximum number of exchanges that a bubble sort algorithm can perform.

The normalized Kendall tau distance values, i.e. the Kendall-tau distance divided by $N \times (N-1)/2$ (possible values ranging from [0..1.0]), is shown in Table 4 and Table 5.

Table 4 Normalized Kendall's tau distance for C SCG.

	$R_C^{(1)}$	$R_C^{(2)}$	$R_C^{(3)}$	$R_C^{(4)}$	P_C
$R_C^{(1)}$	0.000				
$R_C^{(2)}$	0.208	0.000			
$R_C^{(3)}$	0.217	0.008	0.000		
$R_C^{(4)}$	0.183	0.025	0.033	0.000	
P_C	0.379	0.358	0.367	0.367	0.000

Table 5 Normalized Kendall's tau distance for C++ SCG.

	$R_{C++}^{(1)}$	$R_{C++}^{(2)}$	$R_{C++}^{(3)}$	$R_{C++}^{(4)}$	P_{C++}
$R_{C++}^{(1)}$	0.000				
$R_{C++}^{(2)}$	0.095	0.000			
$R_{C++}^{(3)}$	0.095	0.000	0.000		
$R_{C++}^{(4)}$	0.095	0.000	0.000	0.000	
P_{C++}	0.300	0.367	0.367	0.367	0.000

5 Discussion

5.1 Secure Coding Guidelines for C

Table 2 shows the results of the ranked secure coding guidelines for metrics 1..4 and for the students, all for the C programming language. In this table, lower numbers mean higher ranks and larger numbers mean lower ranks. For example, based on Metric 1, the top-5 ranked SCG is [STR38, STR31, ARR38, EXP34, MEM30], while using Metric 2 they are [STR38, EXP34, STR31, ARR38, EXP33]. Addionally, Table 4 shows the corresponding Kendall distance between the R-SCG. For example, the distance between the R-SCG using Metric 1 and Metric 3 is 0.217.

In Table 2 we can see that we can group the obtained results into three different clusters: 1:$\{R_C^{(1)}\}$, 2:$\{R_C^{(2)}, R_C^{(3)}, R_C^{(4)}\}$ and 3:$\{P_C\}$ according to their relative distances. The 3rd cluster is the one that is most distant from all the other clusters, with a distance bigger in the range $]0.35, 0.38[$. Since this cluster represents the feedback given by students, it also means that their answers are the most farther away from our outcome using real-world data. Furthermore, the 1st cluster (Metric 1) is also distant from the 2nd cluster (Metric 2, 3 and 4), whereby the normalized Kendall-tau distance is bigger than $]0.22, 0.25[$. It is surprising that the Metric 2, 3 and 4 have low distance and form a separate cluster to Metric 1. This discrepancy is most likely due to the the fact that the first metric, since it takes the average CVSS score, tends to lower the overall metric value, while all the other metrics penalize on higher CVSS scores, potentially leading to a different sorting of the list. It is nonetheless interesting to note that, for the defined four metrics, the list of the top-4 most important SCG still contain the same guidelines.

The secure coding guidelines which has gotten the highest ranking among the students was FIO30-C, which is "exclude user input from format strings". The same guideline is ranked lower using Metric 1, 2, 3 and 4, having ranks 7, 6, 7, 5 respectively. Although

not following this SCG can obviously lead to vulnerabilities, in order to exploit it, several additional conditions must be met - this is reflected, in practice, by the lower rank achieved by the results based on real-world data.

The lowest normalized Kendall-tau distance is between Metric 2 and Metric 3. This can also be seen in Table 4, where only R-SCG with Rank 6 and 7 are swapped.

5.2 Secure Coding Guidelines for C++

Table 3 shows the results for the C++ programming language of the ranked secure coding guidelines for metrics 1..4 and from the students' input. In this table, lower numbers mean higher ranks and larger numbers mean lower ranks. Table 5 shows the corresponding Kendall distance between the ranked lists.

Same as for the C secure coding guidelines, we can group the results into three clusters 1:$\{R^{(1)}_{\text{C++}}\}$, 2:$\{R^{(2)}_{\text{C++}}, R^{(3)}_{\text{C++}}, R^{(4)}_{\text{C++}}\}$ and 3:$\{P_{\text{C++}}\}$ according to their relative distances. For this case, the following results are immediately apparent: (1) the clusters are the same as for the C programming language, (2) there are three values which have the zero distance (i.e. are the same) and (3) the distance $\{P_{\text{C++}}\}$ to the other ranked lists is about the same as for the C R-SCG.

For the first observation, this re-states that the metrics 2, 3 and 4 do not produce significantly different results, as for the C SCG case. The second observation means that, for the C++ case, the metrics have lead to exactly the same R-SCG results (i.e. the same ranked list). The third observation means that the students, as for the C R-SCG, have a different perception for what is important as what was extracted from real-world data.

Furthermore, we see that the Top-7 R-SCG for C++ are all the same, independently of using Metric 1, 2, 3 or 4. The same cannot be said for the ranking obtained from the students.

5.3 Threat to Validity

We can see the following possible sources of threats to the validity of this work.

1. We have selected four metrics. However, we might have missed the definition of a metric that leads to very different results (maybe even close to the Students' ranking). However, our experience in the field tells us that the metrics hereby defined and the results obtained are consistent with what has been observed in practice.
2. Only 34 students have been involved in the questionnaire and the statistical results might differ if we increase the population size.
3. We have recurred to cybersecurity experts to untie SCG which had the same metric value. Holm et al. [14], and Allodi et al [1] discuss possible discrepancies that expert opinion might add to the scoring. Nevertheless, the results hereby presented have shown that the Kendall distance is not too much sensitive on the values of the scores.
4. Our work did not consider the impact of the standard deviation of the SCG metric. Taking this into consideration, the Kendall distance between the participants answers and the computed rankings could change and also lead to different conclusions.
5. This work did not consider SCG that can be checked with an automatic tool, such as static code analysis. However, our experience is that it is not sufficient to use tools, the developers should also know how to interpret their results. This is only possible with training and awareness.

6 Conclusions and Further Work

In an industrial context, training of software developers in secure coding is a costly activity that needs careful thought and planning. Software bugs often result in vulnerabilities which, when exploited, can lead to serious damage. Secure coding guidelines (SCG) exist nowadays in order to educate software developers and make them aware on how to avoid writing such bugs. However, it has been previously shown that not all software developers are knowledgeable on the said secure coding guidelines. Combined with the restrictions from the industry, this paper proposes a method to rank secure coding guidelines which in turn can be used to prioritize the training of software developers on the SCG. By focusing attention and addressing the most important secure coding guidelines(i.e. the ones that cause the most impact) first, a trained software developer can avoid the major problems and software bugs that have been plaguing the industry.

The major contribution of this work are two sets of ranked secure coding guidelines, one for C and another for C++. Another contribution of this work is the comparison of the gap between industry relevant ranking and ranking from academic students. Here we also see that, although academic students might even be well trained in writing secure code, their ranking of SCG did not match what is obtained from real-world data. As further work we would like to evaluate how and if the usage of automated tools (e.g. static code analysis) influences the results hereby presented.

References

1 Luca Allodi, Sebastian Banescu, Henning Femmer, and Kristian Beckers. Identifying relevant information cues for vulnerability assessment using cvss. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, CODASPY '18, pages 119–126, New York, NY, USA, 2018. ACM. doi:10.1145/3176258.3176340.

2 Motor Industry Software Reliability Association. Guidelines for the use of the c language in critical systems. Standard, Motor Industry Software Reliability Association, Nuneaton, Warwickshire, UK, March 2012.

3 Motor Industry Software Reliability Association. Additional security guidelines for misra c:2012. Standard, Motor Industry Software Reliability Association, Nuneaton, Warwickshire, UK, March 2016.

4 Roberto Bagnara, Abramo Bagnara, and Patricia Hill. The MISRA C coding standard and its role in the development and analysis of safety- and security-critical embedded software. *CoRR*, abs/1809.00821, 2018. arXiv:1809.00821.

5 International Electrotechnical Commission. Industrial communication networks - network and system security - part 2-1: Establishing an industrial automation and control system security program. Standard, International Electrotechnical Commission, October 2010.

6 International Electrotechnical Commission. Security for industrial automation and control systems - part 4-1: Secure product development lifecycle requirements. Standard, International Electrotechnical Commission, January 2018.

7 Felix Fischer, Konstantin Böttinger, Huang Xiao, Christian Stransky, Yasemin Acar, Michael Backes, and Sascha Fahl. Stack overflow considered harmful? the impact of copy&paste on android application security. In *IEEE Symposium on Security and Privacy*, pages 121–136, San Jose, CA, USA, 2017. IEEE Computer Society. doi:10.1109/SP.2017.31.

8 Software Assurance Forum for Excellence in Code. Safecode – fundamental practices for secure software development – essential elements of a secure development life-cycle program, 3rd ed. Standard, NIST, March 2018.

9 Pascal Gadient, Mohammad Ghafari, Patrick Frischknecht, and Oscar Nierstrasz. Security code smells in android ICC. *CoRR*, abs/1811.12713:3046–3076, 2018. arXiv:1811.12713.

10 Tiago Gasiba, Kristian Beckers, Santiago Suppan, and Filip Rezabek. On the requirements for serious games geared towards software developers in the industry. In Daniela E. Damian, Anna Perini, and Seok-Won Lee, editors, *27th IEEE International Requirements Engineering Conference, RE 2019, Jeju Island, Korea (South), September 23-27, 2019*. IEEE, 2019. URL: https://ieeexplore.ieee.org/xpl/conhome/8910334/proceeding.

11 John Goodall, Hassan Radwan, and Lenny Halseth. Visual analysis of code security. In *Proceedings of the Seventh International Symposium on Visualization for Cyber Security*, VizSec '10, pages 46–51, New York, NY, USA, 2010. ACM. doi:10.1145/1850795.1850800.

12 Katerina Goseva-Popstojanova and Andrei Perhinschi. On the capability of static code analysis to detect security vulnerabilities. *Inf. Softw. Technol.*, 68(C):18–33, December 2015. doi:10.1016/j.infsof.2015.08.002.

13 Norman Hansch and Zinaida Benenson. Specifying it security awareness. In *25th International Workshop on Database and Expert Systems Applications, Munich, Germany*, pages 326–330, September 2014. doi:10.1109/DEXA.2014.71.

14 Hannes Holm and Khalid Afridi. An expert-based investigation of the common vulnerability scoring system. *Computers & Security*, 53, May 2015. doi:10.1016/j.cose.2015.04.012.

15 Siv Houmb, Virginia Franqueira, and Erlend Engum. Quantifying security risk level from cvss estimates of frequency and impact. *Journal of Systems and Software*, 83:1622–1634, September 2010. doi:10.1016/j.jss.2009.08.023.

16 Hongyi Hu, Bryan Eastes, and Michelle Mazurek. Toward a field study on the impact of hacking competitions on secure development. In *The 4th Workshop on Security Information Workers Baltimore Marriott Waterfront*, Baltimore, MD, USA, August 2018.

17 Russell Jones and Abhinav Rastogi. Secure coding: Building security into the software development life cycle. *Information Systems Security*, 13(5):29–39, 2004.

18 Ankur Joshi, Saket Kale, Satish Chandel, and Dinesh Pal. Likert scale: Explored and explained. *British Journal of Applied Science & Technology*, 7:396–403, January 2015. doi:10.9734/BJAST/2015/14975.

19 Maurice Kendall. A New Measure of Rank Correlation. *Biometrika*, 30(1-2):81–93, June 1938. doi:10.1093/biomet/30.1-2.81.

20 Ryo Kurachi, Hiroaki Takada, Masato Tanabe, Jun Anzai, Kentaro Takei, Takaaki Iinuma, Manabu Maeda, and Hideki Matsushima. Improving secure coding rules for automotive software by using a vulnerability database. In *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 1–8, September 2018. doi:10.1109/ICVES.2018.8519496.

21 Na Meng, Stefan Nagy, Danfeng Daphne Yao, Wenjie Zhuang, and Gustavo Arango. Secure coding practices in java: challenges and vulnerabilities. In *IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 372–383, May 2018. doi:10.1145/3180155.3180201.

22 MITRE-Corporation. Common weaknesses enumeration, 2019. URL: https://cwe.mitre.org/.

23 MITRE-Corporation. CVE details, 2019. URL: https://www.cvedetails.com/.

24 Alena Naiakshina, Anastasia Danilova, Christian Tiefenau, and Matthew Smith. Deception task design in developer password studies: Exploring a student sample. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, pages 297–313, Baltimore, MD, 2018. USENIX Association. URL: https://www.usenix.org/conference/soups2018/presentation/naiakshina.

25 National Institute of Standards and Technology. Nist special publication 800-37, guide for applying the risk management framework to federal information systems a security life cycle approach. Standard, NIST, February 2010.

26 International Standards Organization. ISO/IEC 27002:2013 – Information technology – Security techniques – Code of practice for information security controls. Standard, International Standards Organization, October 2013.

27 Blerim Rexha, Arbnor Halili, Korab Rrmoku, and Dren Imeraj. Impact of secure programming on web application vulnerabilities. In *2015 IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS)*, pages 61–66, November 2015. doi:10.1109/CGVIS.2015.7449894.

28 IEEE Spectrum. The Top Programming Languages 2018. https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2018, 2019. [Online; accessed 27-October-2019].

29 PCI SSC. Payment Card Industry – Payment Application Data Security Standard – Requiremenst and Security Assessment Procedures, v3.1. Standard, PCI SSC, May 2015.

30 Madiha Tabassum, Stacey Watson, Bill Chu, and Heather Richter Lipford. Evaluating two methods for integrating secure programming education. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE 2018, Baltimore, MD, USA, February 21-24, 2018*, pages 390–395, 2018. doi:10.1145/3159450.3159511.

31 Carnegie Mellon University. Secure Coding Standards. https://wiki.sei.cmu.edu/confluence/display/seccode, 2019. [Online; accessed 19-March-2019].

32 Michael Whitney, Heather Richter Lipford, Bill Chu, and Tyler Thomas. Embedding secure coding instruction into the ide: Complementing early and intermediate cs courses with eside. *Journal of Educational Computing Research*, 56:073563311770881, May 2017. doi:10.1177/0735633117708816.

33 Xin-Li Yang, David Lo, Xin Xia, Zhi-Yuan Wan, and Jian-Ling Sun. What security questions do developers ask? a large-scale study of stack overflow posts. *Journal of Computer Science and Technology*, 31(5):910–924, September 2016. doi:10.1007/s11390-016-1672-0.

34 Tianyi Zhang, Ganesha Upadhyaya, Anastasia Reinhardt, Hridesh Rajan, and Miryung Kim. Are code examples on an online q&a forum reliable?: A study of api misuse on stack overflow. In *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, pages 886–896, New York, NY, USA, 2018. ACM. doi:10.1145/3180155.3180260.

An Arduino Simulator in Classroom – a Case Study

Paulo F. Gonçalves
Coimbra Polytechnic - ISEC, Portugal
a21171940@isec.pt

João Sá
Escola Secundária de Avelar Brotero, Coimbra, Portugal
joaosa@gmail.com

Anabela Coelho
Agrupamento de Escolas de Pombal, Portugal
anabela.coelho@aepombal.edu.pt

João Durães
Coimbra Polytechnic - ISEC, Portugal
jduraes@isec.pt

Abstract

The Arduino Platform is increasingly being used as a central component in introductory programming courses of the curricula in middle, high school and even higher education. Given this scenario it is pertinent to understand how the cost-effectiveness, reliability and accessibility of this central component can be improved. We propose the use of an Arduino simulator to improve usability, cost, and class efficiency, allowing for improved and even new forms of use and course benefits. This paper presents and describes an Arduino simulator that we developed for education purposes, and a case study of its use in embedded programming courses from two high-schools. We compared its use against the usual use of real hardware platform analyzing usability, student workload and time efficiency. Our results, that we present and discuss, suggest that there are no apparent drawbacks in using the simulator, and some metrics such as basic exercise-solving efficiency and global effort showed an improvement.

2012 ACM Subject Classification Applied computing → Computer-assisted instruction

Keywords and phrases Arduino, Education, Simulator

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.12

1 Introduction

Teaching programming with microcontrollers is becoming increasingly common in middle and high-schools, even outside electronics courses. Several factors promote this scenario: digital literacy is now viewed as an important aspect of enabled citizenship that should be promoted as early as possible [11, 14], the increasing pervasiveness of the *Internet of Things* (IoT), and the low cost of microcontrollers when compared to traditional desktop computers making them an interesting tool for programming courses.

One of the platforms most commonly used and best adapted to the learning with microcontrollers scenario is the Arduino Platform [2] since it combines the simplicity, yet resourcefulness, of hardware with the easiness of an integrated development environment well suited for people without significant knowledge on microcontrollers [6], and has the additional advantage of being an *open source* platform which means no licensing costs. It is then no surprise that many middle and high-schools are now including subjects of programming using Arduino [1, 12, 10].

First International Computer Programming Education Conference (ICPEC 2020).
Editors: Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões; Article No. 12; pp. 12:1–12:12

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Despite its many positive features, the use of the Arduino platform in classroom has several aspects that may decrease the efficiency of the education effort (e.g., cost and classroom time efficiency), and may even prevent the use of some forms of learning (e.g., distance learning), as detailed in the next section. We perceive these aspects as an opportunity for improvement by introducing the use of a *virtual Arduino*, that is, an Arduino Simulator that maintains all the advantages of the real hardware, decreases or removes the aspects that decrease class performance, and open new avenues and forms of learning not present with the real Arduino.

The use of an Arduino simulator in classroom is not common. In fact, to the best of our knowledge, no school in our country uses one, and fully or at least usable Arduino simulators in teaching context are not available. We propose to contribute to this scenario by developing and making freely available a simulator that can be used in classroom allowing the development and testing of programs and small circuits in a way that is similar to the real hardware platform, and is compatible with the use of the same IDE in the same way as with the real hardware, maintaining the same procedures the developer is used to.

This paper presents the several aspects relevant to our simulator and its use: in the next section we detail the aspects that lead to the opportunity and advantages of using a simulated Arduino in classroom, leading to high-level requirements and goals for the simulator. The overall architecture and technical overview of the simulator implementation is presented in the following section. Next we describe a case study of its use in the context of two schools and analyze the results concerning its usability and advantages. We conclude the paper with a summary of our results concerning the use of the simulator in classroom.

2 Motivation and goals

Arduinos are a key component of computing resource in the classroom, providing the computing power needed to run small experiments, allowing pure programming exercises, and enabling a first contact with electronic devices. Given its advantages, the tendency to use this device in classroom will probably continue and possibly even increase. However, the use of Arduinos in classroom presents some aspects that can be improved and opportunities that can be explored to further advance the learning process. We analyze these next.

2.1 Limitations of actual hardware use

We perceive the following limitations when using the Arduino in classroom; device wear-down, cost, skill mix-up, manual dexterity, assembly issues, time-efficiency.

Device wear-down and cost. The average number of writes of an Arduino's flash memory before failure is about 10,000 times [4]. This number is easily reached in just two years of use in classroom. If we consider the common scenario of one school having 6 classes interacting with 1 Arduino-equipped room, with 5 exercises per lesson, 10 tries per exercise (low estimation), and 15 lessons per year, we will have about 4500 writes per year. This will cause the school having to replace its Arduinos every two years.

Skill mix-up. Typical Arduino setups involve some electronic (LEDs, resistors, etc.), introducing the need for basic electronics skills, which are not typical for young students and may act against the goals of learning programming [7].

Manual dexterity. The small components used with Arduino projects require some level of fine movement control. Students with slight muscle-control disorder will find it very difficult to assemble the circuits. Schools should be inclusive and even if the number of affected students is small, this is an issue to consider.

Assembly issues. Sometimes the components' connectors are not in perfect condition and will not work well causing the project to fail by reasons outside the immediate control of the student, triggering frustration and wasting time.

Lesson time-efficiency. Arduino projects require some time to collect all the components, sorting and setting them up, and at the end of the lesson, gathering and storing. When compared to the typical lesson duration, this may add up to a considerable amount of time not being used as actual learning time.

2.2 Mitigation offered by simulation

We propose to mitigate these issues with a simulator in the following manner:

Device wear-down and cost. The components used with the simulator are virtual. There is no wear-down and no need for periodical replacement. This will immediately bring down the cost of maintaining a room equipped with Arduinos, as the computers used for the simulator typically are already available.

Skill mix-up. The electronic skill requirement can be reduced or even entirely removed if the simulator focus more on the logical aspects of the components and alleviate unnecessary details (e.g., simulated LEDs do not require an extra resistor), enabling students to focus on the central aspects of programming.

Manual dexterity and Assembly issues. Using a simulator with graphical interface and moving a pointing device is easier than handling small components. This removes some impediments to student having fine-grain movement disorders. It also solves the issues related to defective components and assembly problems.

Lesson time-efficiency. Setting up a simulated project can be as simple and fast as turning a computer on, executing a program and opening the project file. Sorting and placing components is replaced by point, click and drag components in the screen. This is faster than working with physical components, making more time available for learning.

2.3 New opportunities made available by simulation

Using a simulated Arduino opens new opportunities that can greatly improve the learning outcomes. We list those more immediately relevant:

Distance learning. Students typically don't take the hardware home. Students that cannot attend one lesson will loose that lesson; students wishing to improve skills on their own time will not be able to do so and will have to wait for the next lesson. A simulator can solve this limitation by allowing the student to use the simulator software at home, either running the simulator on his personal desktop, or by connecting to a simulator hosted on a server at school.

Project continuation support. Larger projects that cannot be concluded in one lesson are dismantled to reuse its components, preventing its continuation on the next lesson. A simulated project is just information that can be stored in a file and later reopened for continuation. This opens an entirely new possibility for larger projects.

Debugging. Debugging is not directly available on real Arduino hardware due to hardware constraints. However, debugging is important and should be encouraged. Simulators do not share the constraints of the real hardware and can offer the means to debug the code, including advanced functionality such as step-by-step execution and memory inspection.

We could identify more new uses made available by simulation. However, we see these as the most immediately relevant. We focused on these first and our simulator already supports them.

2.4 High level requirements and goals

The following are a set of high-level requirements that we identified to address the limitations and provide for the opportunities listed above. These requirements guided the simulator architecture definition and implementation choices.

Compatibility. To minimize or even remove intrusion and foreign aspects considering the typical real-Arduino development setup, the simulator must present itself as just another type of board and all it is required is to use the same IDE and simply configure this new board type. All the development and code upload to the (simulated) Arduino is then carried out in the same fashion as with real Arduinos.

Client-server architecture. To provide for easy central management, reduce operational and maintenance complexity, and allow access to users with minimum local setup, the simulator is hosted in a server where the actual simulation takes place. The user interacts with the simulation in two ways: via the usual IDE to develop the code, and via the client to interact with the circuit and components.

Web-compatibility. We decided that interaction with the simulator should be made via a web-browser to maximize usability. By using a web-based interface and protocols, we also gain the accessibility provided by the web.

3 Context and related work

3.1 Simulator implementations

There are several simulators available. We analysed their characteristics considering our goals (see summary in Table 1):

Binary-level code compatibility. This is important and needed to maintain the program loading process the same as with the read hardware. Our survey shows that more than half of the existing simulators are compatible only at the API level, meaning that they only simulate a fixed known basic functions of the Arduino, simulating their operation but not the code execution itself. This may prevent many existing libraries for Arduino from running in the simulator.

IDE compatibility. It is also important to minimize changes in the environment the developer is used to. None of the simulators we analysed is compatible with the Arduino IDE. This is contrary to the notion that in a teaching scenario, students should learn using the same tools that they will later use.

Web accessibility. We previously identified web-based interface as one of the requirements desirable for the simulator. However, none of the simulators that are binary-level compatible provide a web interface.

Debug ability. Although several of the simulators that are binary compatible allow debugging, all except one require an external tool, and the exception to this is not freely available. This either increases complexity and removes compatibility with the IDE, or causes extra costs.

3.2 Simulation/virtualization techniques

There are three main virtualization techniques: interpretation, compiled simulation and dynamic translation. The latter transforms instructions from the target architecture (Arduino microcontroller) to the host architecture (of the machine running the simulator), in practice, rewriting the code. This type of transformation may insert undesired effects in timing and

Table 1 Arduino simulators comparison.

Software	Free	Open-source	Cross-platform	Web interface	binary compatible	Maintained	Allows Debug	Arduino IDE
Proteus (https://www.labcenter.com/)	No	No	Yes	No	Binary	Yes	Yes	No
Virtronics Simulator for Arduino (https://virtronics.com.au/Simulator-for-Arduino.html)	No	No	No	No	API	No	Yes	No
VBB4Arduino (http://www.virtualbreadboard.com/)	No	No	No	No	API	Yes	Yes	No
123D Circuits (https://123d.circuits.io/)	Yes	No	Yes	Yes	API	No	–	No
ArduinoDebugger (https://github.com/Paulware/ArduinoDebugger)	Yes	Yes	No	No	API	No	Yes	No
CodeBlocks Arduino IDE (http://arduinodev.com/codeblocks/)	Yes	Yes	No	No	API	Yes	–	No
Simuino (http://web.simuino.com/home-1)	Yes	Yes	Yes	Yes	API	No	Yes	No
Emulino (https://github.com/ghewgill/emulino)	Yes	Yes	Yes	No	Binary	No	No	No
Atmel Studio 7 (https://www.microchip.com/mplab/avr-support/atmel-studio-7)	Yes	No	Yes	No	Binary	Yes	Yes[1]	No
Emulare (http://emulare.sourceforge.net/)	Yes	Yes	Yes	No	Binary	No	Yes[2]	No
SimAVR (https://github.com/buserror/simavr)	Yes	Yes	Yes	No	Binary	Yes	Yes[2]	No

controllabillity that may affect the intended behavior of the original code and prevents the debugging ability. Interpretation technique simulates each instruction, one by one, while compiled simulation builds an entire program in the host architecture with the instructions needed to simulate the complete sequence of the simulated instructions. Interpretation offers more controllability to support debugging, but in theory is slower than compiled simulation. We conducted a study to compare the performance of these two techniques [8] and concluded that, in our case, using Java, interpretation is faster. Thus, we opted for the interpretation technique to implement the simulator.

3.3 Arduino platform

Arduino is a platform composed by both hardware and software that can be used to control many types of electronic components and projects. The hardware is a printed circuit board with an AVR microcontroller [3], a power supply, a serial interface for programming, input and output connections, and a bootloader to program the device. The software component includes an API and libraries to manipulate the hardware and components, and a self-contained integrated development environment (IDE) to develop and upload code to the device.

4 Architecture and Implementation

The simulator is organized as a typical web-based client/server system. The server hosts the simulation logic and mechanisms and can serve multiple independent simulations at the same time, depending on the computing power. The client handles all the user and IDE interaction. The user interface is based on common web technology and compatible with common web-browsers. Figure 1 depicts the modules composing the simulator, which are described next.

[1] with extra hardware
[2] with external debugger

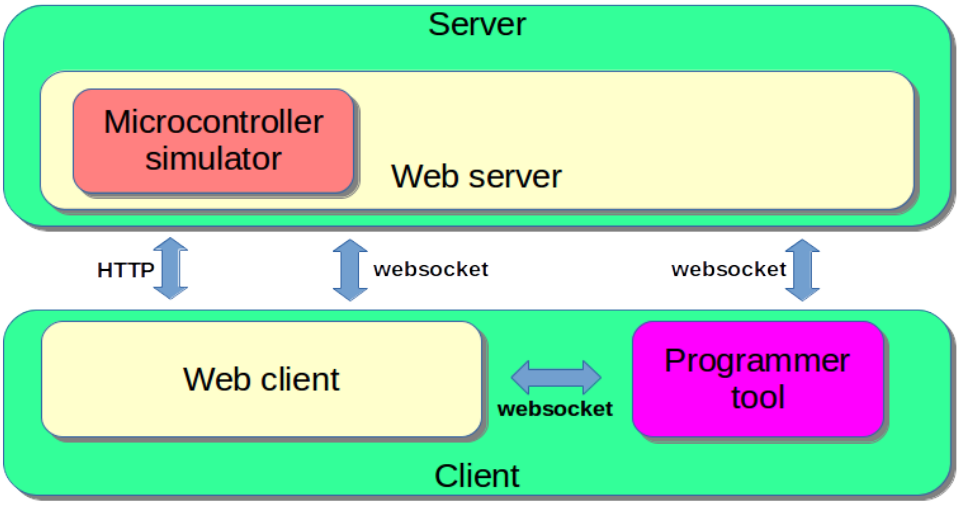


Figure 1 Architecture.

Microcontroller simulator. It is in this module that all the features of the microcontroller are implemented, namely the AVR Instruction Set, the microcontroller peripherals, FLASH and SRAM. This module executes the microcontroller code, exposes methods to change pins values and throws events when their state is changed. Its internal structure is very modular and each part can be easily replaced by other to allow simulating other Atmel microcontrolers (the one now simulated is the ATmega328P [4]) maintaining the integration with the parts responsible for simulating the ISA, FLASH, SRAM and the peripherals.

Web server. This module manages the users, maintains simulation instances, and links simulations to their respective user client and *programmer tool*. It is also responsible for storing user-created projects and all their related data.

Web client. This is where the user creates circuits to simulate. There is a drawing area available where the user places and connects the Arduino and various electronic components. All the simulator functionality can be accessed with the client: start/pause the simulation, execute step-by-step, inspect FLASH and SRAM memory contents, manage breakpoints, etc. Establishing a relationship between a web-client, the IDE/*programmer tool* and the web server cannot be done directly given the way browser *sandboxing* works. The client periodically sends information to the *programmer tool* to establish this relationship.

Programmer tool. This module corresponds to a board driver that is installed in the Arduino IDE replacing the device programming program (avrdude [5]) so that when uploading the binary code, instead of programming a real device the binary is sent to the simulator in the web server.

The web client interface can be seen on Figure 2, with the toolbar, components palette, drawing area with an example circuit and the inspecting windows for Source Code, FLASH and SRAM opened.

The usage of the Arduino IDE is the same as programming a real Arduino with the exception of selecting the new device installed by the board driver. In Figure 3 is possible to see the IDE with the same program loaded in the simulator.

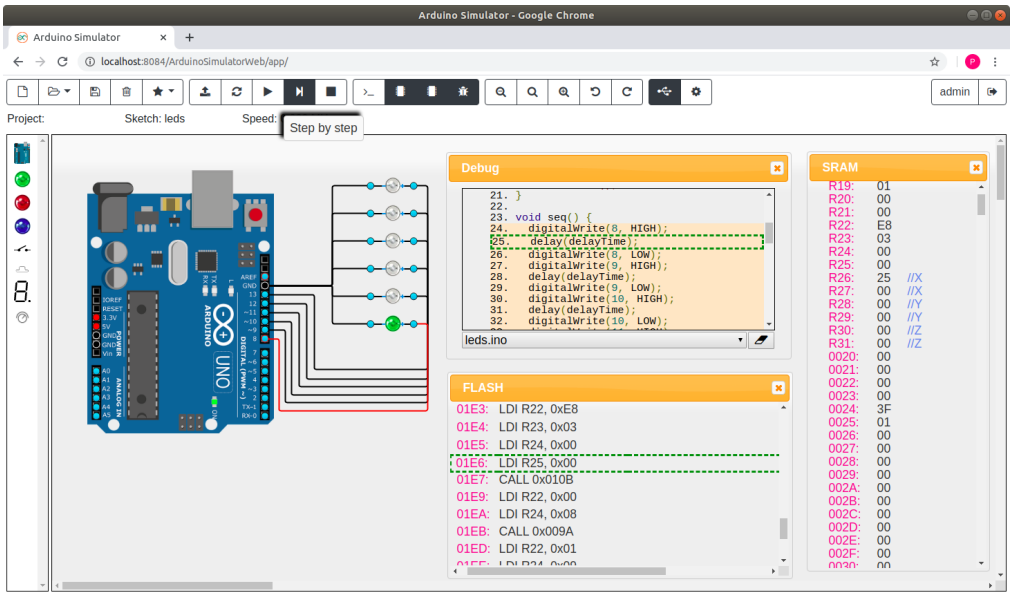


Figure 2 Simulator user interface.

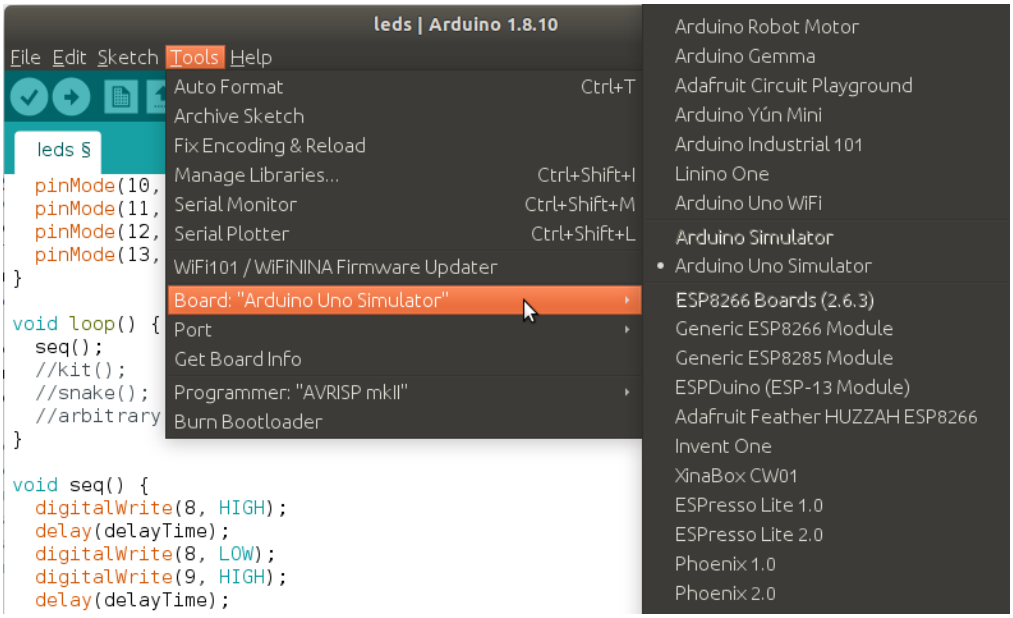


Figure 3 Arduino IDE.

5 Experimental use in real scenario

We conducted a real-use case study in two high-schools of the region to assess the usability and advantages of using the simulator in the classroom. The case study comprises 5 classes, 3 from *Escola Secundária de Avelar Brotero* (Coimbra), and 2 from *Agrupamento de Escolas de Pombal* (Pombal). In both schools students were aged between 16 and 18 years old. Some of the classes belonged to courses of technological area, while other belonged to health. The

Table 2 Characterization of students.

School	Class	Area	Curricular Year	# students	# Sim.	# Real
Avelar Brotero	Class 1	Sciences	12^{th}	29	12	17
Avelar Brotero	Class 2	Health	12^{th}	29	12	17
Avelar Brotero	Class 3	Mixed	12^{th}	20	12	8
Pombal	Class 4	Electronics	11^{th}	11	6	5
Pombal	Class 5	Electronics	12^{th}	12	7	5
			Total:	101	49	52

duration of the lessons was not the same for all tests and to compensate, the number of exercises also varied. The participation in the experiment was optional and we did not notice any reservation from any student. In each class, half the students used real Arduinos and the other half used the simulator. Table 2 summarizes the characteristics of the classes and students. Our methodology is described next.

5.1 Methodology

We separated each class into two groups of students: one used a real Arduino Uno and the other used the simulator. We balanced the groups in terms of experience and knowledge both in programming and in the Arduino Platform. This separation counted on the help of the respective teachers. The exercises presented to the students consisted on programming challenges involving simple circuits and were the same for both groups. These exercises were the usual for those classes, were prepared by the teachers and had no influence or change related to the simulator.

Due to the size of the classes and the lack of computers for all students at the school *ESAB* the exercises were performed in groups of 2 students. This was already the common scenario for those classes and it had nothing to do with the simulator.

We measured the efficiency of the simulator as a learning tool by observing the time students took to solve the exercises, comparing real Arduino with the simulator, the number of exercises completed, and their final result (correct/incorrect). We also used a questionnaire to evaluate the perception of the students about the use of the simulator.

5.2 Exercises

We used three exercises in each test. These exercises were defined by the teachers following their usual plan for the classes and there was no influence in the exercise definition related the Simulator. The exercises had incremental difficulty. All the exercises involved both programming and building a simple circuit. The circuit was assembled on a breadboard or in the simulator client; the code was written with the IDE in all cases.

The exercises were as follows:

1. Blink a LED with one second on and one second off.
2. Make 3 LEDs light up in sequence, ensuring that only one is lit at a time, and with a half-second interval.
3. Flash a set of 3 LEDs intermittently (all at the same time) only when a push button connected to the Arduino is pressed.

5.3 Questionaires

We defined a questionnaire adapted from the NASA Task Load Index (NASA-TLX) [9] which are questionnaires created by the Human Performance Group of the National Aeronautics and Space Administration to assess the workload when accomplishing a given task. This has the dual advantage of considering the point of view of the subjects and including subjective aspects such as discomfort or stress.

NASA-TLX questionnaires consist of 2 parts. In the first, 6 subjective parameters are assessed: *Mental Demand, Physical Demand, Temporal Demand, Performance, Effort* and *Frustration*. Subjects grade each parameter using a scale of 1 (very low) to 20 (very high). This grading is related to the execution of one task and thus, it is repeated for each task. The second part assesses the importance each subject assigns to each parameter and it is given only once at the end of the test. In this part, parameters are compared in pairs and for each pair subjects are asked to identify the parameter most relevant to them. This allows assigning weights to each parameter (for each subject) and compute an overall workload index experienced by each individual. We decided not to use the *effort* parameter since in our context it can be captured individually by the the *physical demand* and *mental demand*. We also adapted the scale from 1-20 to 1-6 to avoid pressuring the students with excessive accuracy when classifying each of the parameters. We introduced additional questions to understand the students background and later assess any possible correlation with the performance shown when using the simulator (Table 5 in next subsection).

5.4 Results and Discussion

After a first assessment of the questionnaires, we noticed that not all students answered all questions, either in the first part or in the second part of the questionnaire and we excluded those from the results. This resulted in a total of 189 valid exercises, 89 of which were performed in the real environment and 100 in the simulator (Table 3).

Table 3 Valid and invalid inquiries.

	Total	Invalid	Valid
Students	101	3	98
Exercises	212	23	189

It should be noted that not all students performed the 3 exercises proposed in class because teachers did not impose a time limit for the exercises (and we did not want to change their usual process) and students only moved on to the next exercise when they finished the previous one. Table 4 shows the number of students that executed each exercise.

In class 1 the students performed all the exercises in a row, having only counted the total time and only responded once to the first part of the questionnaire. However, this happened to both real Arduino and simulator students and the comparison between them remains possible. This was not planned and happened due to insufficient initial communication between the parts involved and was corrected in the following tests.

Figure 4 presents for both real and simulated Arduino the average and median workload index experienced by the students when performing the exercises, and the average and median time they took to solve the exercises.

As we can see, the workload index appears to be approximately the same for both real and simulated Arduino, suggesting that the use of the simulator does not greatly interfere with the overall effort experienced by the students, although when using the simulator it is about 8% lower, which is a positive result towards the use of the simulator.

Table 4 Distribution of exercises across classes.

	Exercise 1		Exercise 2		Exercise 3	
Class	Sim.	Real	Sim.	Real	Sim.	Real
1	8	13	-	-	-	-
2	12	16	12	12	4	3
3	11	8	10	8	8	2
4	5	4	5	4	5	4
5	7	5	7	5	6	5

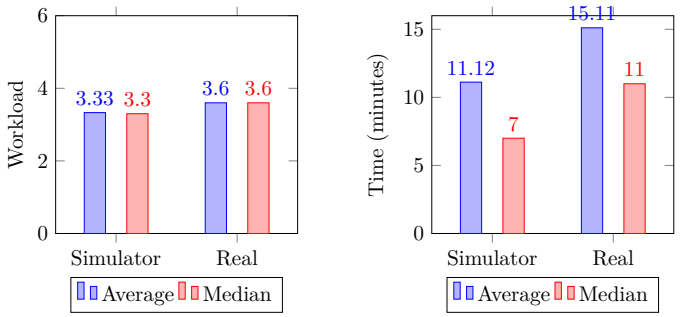


Figure 4 Comparison of workload and exercise time.

Concerning the time parameter, we noticed that students using the simulator take significantly less time to solve the exercises: 26% on average and 36% median. Combined with the lower Physical Demand, this may indicate that using the simulator is more intuitive than making electrical connections on a breadboard. This result suggests that using the simulator is beneficial considering the number of exercises possible to execute during the lesson.

Regarding the Physical Demand (Figure 5), we also observed a significant improvement (one third of the physical demand). We expected an improvement as it is easier to move a mouse than handling small components. The fact that the improvements are so significant is a very encouraging result suggesting that the use of the simulator positively impacts the learning process. Considering Mental Demand and Performance (exercise completion), the results are the same for both groups of students (Figure 5). This was also expected: there was no time limit, so completion depends mostly on the exercise itself; mental performance should also not vary much as the IDE and the programming effort is the same in both cases. This actually is in accordance to the goals of not introducing intrusiveness in the development process.

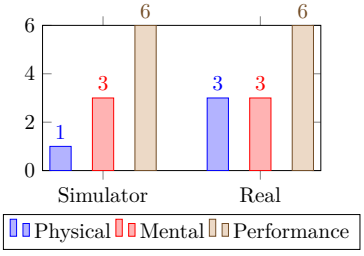


Figure 5 Comparison of the median of Physical Demand, Mental Demand and Performance.

To confirm that the results obtained were not influenced by the students background and previous experience, we analyzed the correlation between the answers about previous background (in the questionnaire) with the workload and the time taken to solve the exercises. We computed the Point Biserial Correlation [13] between the "Yes" answer to the three questions in Table 5 and the average of time to solve the exercises and the average of workload during the exercises.

In the case of previous experience using drawing software and previous experience with Arduino, the correlation is very low, suggesting that these two are not related to the tests results. Concerning the previous experience in programming, the correlation is a little higher but also not significant.

Table 5 Correlation between "Yes" answer, Time and Workload.

Question	Time	Workload
Do you use drawing programs?	−0.099	0.030
Had you already done programming before this school year?	−0.131	−0.158
Have you done programs for Arduino before this course?	−0.047	−0.025

5.5 Teacher point of view

The teacher's point of view is very relevant to our analysis. The teachers of the classes of the case study were involved in all preparation steps and in their opinion, the use of the simulator did not introduced any extra class-management work, and dispensing the handling of components alleviated the beginning and ending of the lesson. There is the need of one initial explanation to the students about the simulator, but that is just for the first lesson using it. So far it seems that the simulator does not involve extra workload to the teachers.

6 Conclusions

Given the increasing use of the Arduino Platform as a key learning tool it is pertinent to address the aspects where this type of use can be improved. We identified a set of aspects where the use of Arduino in classroom can benefit from the use of an Arduino simulator, including new opportunities that can be explored to the mutual benefit of teachers and students. We presented the planning and development of an Arduino simulator that although it can be used for general purpose, it is specifically aimed at its use in education context as its goals and requirements were based on the needs we identified for classroom use. The simulator was implemented in Java and can be run in the typical computers usually found in schools. It has a web-based client-server architecture allowing it to be centrally managed and remotely used, enabling distance learning scenarios. Most importantly, it is compatible with the usual IDE for Arduino and has no impact on the developer usual procedures.

We tested and validated the use of the simulator in classroom in a case-study involving two high-schools comprising five classes using the same exercises already planned by the teachers in regular context. We collected metrics regarding mental and physical workload experienced by the students, and also performance-related metrics such as time spent and exercise completeness. We concluded that the use of the simulator did not have any negative impact on the students or class management, and observed a significant improvement on the physical workload and in the time needed to solve the exercises. This improvement can have a very positive impact on the efficiency of the lesson time, making possible more exercises per

lesson. Concerning the point of view of the teachers, our feedback is that no negative aspects were introduced, class management is easier and all that is needed is one initial explanation to students concerning the use of the simulator.

We analysed the correlation of the background of the students with the results obtained. We did not find any significant correlation and we assume that the performance improvements we observed are indeed related to the use of the simulator, suggesting that its use in classroom is beneficial for learning goals.

We identify several avenues for future work: the continued enhancement of the simulator to pursue further functionality and enable new potential, the continuation of tests in more classroom-related scenarios, and the opening of the simulator as an open-source project [1] to increase its visibility and use. We believe that this simulator is a positive contribution to promote early and broadened digital-literacy and we will look for and pursue any opportunities of partnership with education officials and state-sponsored projects to disseminate the simulator in schools.

References

1 Francesca Agatolio and Michele Moro. A workshop to promote arduino-based robots as wide spectrum learning support tools. In *Robotics in Education*, pages 113–125. Springer, 2017.

2 Arduino.cc. Arduino - home, 2018. URL: `https://www.arduino.cc/`.

3 Atmel. AVR Instruction Set Manual, 2016.

4 Atmel. ATmega328/P Datasheet, 2018.

5 Avrdude. Avr downloader/uploader, 2019. URL: `https://www.nongnu.org/avrdude/`.

6 Hernando Barragán. Wiring: Prototyping physical interaction design. *Interaction Design Institute, Ivrea, Italy*, 2004.

7 Edward Currie and Carl James-Reynolds. The use of physical artefacts in undergraduate computer science teaching. In *E-Learning, E-Education, and Online Training*, pages 119–124. Springer, 2017.

8 Paulo F. Gonçalves, Jorge Bernardino, and João Durães. Virtualization technologies for arduino simulation. In *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, June 2019. `doi:10.23919/cisti.2019.8760727`.

9 Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.

10 Peter Jamieson. Arduino for teaching embedded systems. are computer scientists and engineering educators missing the boat? In *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2011.

11 Marc Prensky. Programming is the new literacy. *Edutopia magazine*, 2008.

12 John Sarik and Ioannis Kymissis. Lab kits using the arduino prototyping platform. In *2010 IEEE Frontiers in Education Conference (FIE)*, pages T3C–1. IEEE, 2010.

13 Robert F Tate. Correlation between a discrete and a continuous variable. point-biserial correlation. *The Annals of mathematical statistics*, 25(3):603–607, 1954.

14 Annette Vee. Understanding computer programming as a literacy. *Literacy in Composition Studies*, 1(2):42–64, 2013. `doi:10.21623/1.1.2.4`.

[1] `https://github.com/pafgoncalves/ArduinoSimulator/`

Benefits of Cloud Services in Education: A Perspective of Database System Students

Gustavo Gutiérrez-Carreón
Universidad Michoacana de San Nicolás de Hidalgo, Morelia, Michoacán, México
http://www.umich.mx
gagutc@umich.mx

Abstract

Currently, there is a growing trend in the use of cloud-based services to support education. The importance of these services is that they are publicly available, allowing students to access these distributed resources most transparently. In this work, a model of satisfactory learning measurement is proposed to analyze the benefits, from the students' perspective, of cloud services related to education. A case study performed in a Database Systems course is presented; in this, under-graduate students can remotely manage database systems using cloud services. The benefits of an online access scheme compared to those of traditional database access are measured in terms of usability and the principles of cognitive load theory.

2012 ACM Subject Classification Applied computing → Interactive learning environments

Keywords and phrases Cloud Computing, Satisfactory Learning, Database Systems

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.13

1 Introduction

The identified trends in the use of cloud computing in education are clear, ranging from the design of cloud-oriented learning environments for future information technology specialists to the training of information technology specialists to enable them to obtain competencies in the use of cloud technologies [9]. Cloud computing is a distributed computing paradigm, where, instead of acquiring information technology products, users access shared resources under various service models through a net-work, usually the Internet [5]. In universities, cloud computing technology, and the construction of learning management platforms improve the rate of resource utilization for teaching processes [6]. It is essential for an educational organization, with its budgetary constraints and sustainability challenges, to use the most appropriate cloud training for a teaching activity. In [1] there is a comparative analysis of how cloud computing technology can be used in e-Learning systems in favor of higher education. The results demonstrate that in addition to cost, there are technical benefits of using Cloud computing, such as, customer's preferred operating systems, stable virtu-al machines, fully redundant architecture, security firewalls, flexibility to meet new requirements, elasticity in new requirements and flexibility in design, among others.

In the case of traditional database system courses, it is essential to highlight that for the delivery of the course, a set of infrastructure software and tools a required, which often must be installed on servers in the institution or on the students' computers. Technically, the benefits of choosing cloud services, that can be accessed remotely, could be related to the reduction of management and maintenance responsibilities. In that sense, there are efforts like DBLearn [10], a Web-based adaptive e-learning system designed especially for database and SQL related courses. The DBLearn system provides several advantages over current systems, solving the major problem of teaching a database course to students of different learning styles and knowledge levels.

Despite the efforts made in related research work, it is still unclear how cloud computing-based services can benefit to education, and, particularly to database systems students. To achieve this goal, we establish the following research question:

- Is there a significant benefit in terms of satisfactory learning of students who use cloud services for database systems compared to students who locally access soft-ware for the same purpose?

To evaluate these benefits, we focus on two types of indicators; on the one hand, those related to the usability of systems [8] and, on the other hand, the theory of cognitive load [12]. Regarding usability, it allows determining the extent to which users can fulfill a task in a satisfactory, effective, and efficient way, emphasizing the context in which they operate, and the specific tasks performed. For cognitive load theory (CLT), primary knowledge consists of generic cognitive skills and cannot be taught because they are acquired unconsciously.

The model proposed in this work is focused on measuring how satisfactory is a learning environment based on cloud services for database system students. Subsequently, a case study is presented, in which a group of university students is evaluated, which are divided into two groups: a control group and an experimental group. The control group will be asked to carry out a set of activities that involve the use of traditional tools for database system courses. The experimental group will be asked to perform the same set of activities but under an online environment using cloud services for database systems. Finally, the results and conclusions are presented and analyzed.

2 Benefits of a cloud-based solution

Cloud computing providers offer their services according to three fundamental mod-els: Infrastructure as a service (IaaS), platform as a service (PaaS), and Software as a service (SaaS) [6]. Every day there is a growing list of critical applications that are implemented and consumed through cloud services under the Software as a Service (SaaS) mechanisms. The following are some of the benefits of implementing a cloud-based solution [11]:

- The cloud model is efficient and profitable, where the user pays only for re-sources he consumes.
- There are no problems due to the maintenance and updating of the infrastructure and physical hardware.
- The cloud service provider offers autoscaling capabilities.
- Allows you to work from anywhere at any time and from any device
- Accelerates application development.

Despite these benefits, research needs to be done on the impact that cloud architecture has on student learning. In this sense, in the next section we will present a satisfactory learning measurement model, understood satisfaction, as a measure that can provide information on the effectiveness of instructional design.

3 A satisfactory learning measurement model

The satisfactory learning measurement model is a proposal developed by Bradford [3], who detected a correlation coefficient between satisfaction and cognitive load separating academic performance. In this sense, the model focuses on the measurement of three indicators to reduce the cognitive load: awareness, challenge, and commitment.

To measure usability, we take an instrument developed by [4], called the system usability scale (SUS). SUS is a survey consisting of 10 questions that are measured on a simple 5-point Likert scale (ranging from 1-totally disagree 'to 5-totally agree), which offers a global view of subjective assessments of usability. Table 1 shows the questions that are part of the SUS survey.

Table 1 SUS questions.

Q1	I think I would like to use this system frequently.
Q2	I found the system unnecessarily complicated.
Q3	I think the system is easy to use.
Q4	I think I need the support of a technical person to use this system.
Q5	I found that the various functions in this system were well integrated.
Q6	I think there are too many inconsistencies in this system.
Q7	I imagine that most people would learn to use this system very quickly.
Q8	I found the system very complicated to use.
Q9	I felt very safe using the system.
Q10	I needed to learn many things before I could start using this system.

In the case of cognitive load, an instrument proposed by [7] is used. Regarding the awareness indicator, four questions are posed to examine whether the curriculum and assignment instructions were clear (questions 11-14). Regarding the challenge indicator, three questions were established to relate the degree of student satisfaction with the degree of challenge they face (questions 15-17). Regarding the commitment indicator, three questions were asked to relate the relevance of the different types of learning activities to the needs and objectives of the students (questions 18-20). Table 2 presents the ten questions used in the cognitive load - satisfaction questionnaire.

Table 2 Cognitive Load - Satisfaction questions.

Q11	I think the instructions and guidelines for experimenting were clear.
Q12	I think it was understood how the solution to the problem is found, the expected results, and the evaluation process.
Q13	I think the informative sessions and the material presented helped me solve the problem.
Q14	I think this activity will be useful for the development of fu ture projects with the tools presented.
Q15	I believe that the development of this activity challenges my abilities to solve these types of problems.
Q16	I think that solving the problem itself is a significant achievement.
Q17	I can solve this problem and others, more complicated, of the same type.
Q18	I think this experiment is relevant to the course and my curriculum.
Q19	I believe that communication, discussions, or debates with my classmates and the teacher are essential.
Q20	I think this type of activity encourages me to develop solutions for me.

4 Design of the experiment

The experiment was carried out with third-year university students (N = 80) of the Relational Databases course for the program of Technology Manager degree, which contains the classic content of an introductory database course for non-computer science students: entity-relationship model, relational model, design of relational databases, structured query language, transaction control, and concurrency and per-form backup and recovery. The students who participated in the experience were divided into two groups: an experimental group (N = 40) and a control group (N = 40). The same teacher instructed both groups of students.

The experimental procedure consist in twenty 60-minute sessions were conducted for over six months. The first five sessions were for installing a MySQL platform with a database manager. The control group had to install personal laptops with MySQL Workbench database manager [2]. The experimental group create a Google Cloud account (1-year free account with $300 to use) and create instances of Cloud SQL with the PhPMyAdmin database manager [11]. The rest of the fifteen sessions, both groups, manage databases, execute SQL queries and backup and restore databases. After the end of the experiment, all students answered the usability and cognitive load questionnaires.

5 Presentation and importance of the results

The results of the usability questionnaire, both for the control group and for the experimental group, are shown in Figure 1 with a density diagram. In the case of questions 2,4,6,8 that are expressed negatively, the adjustment has been made so that they can be visualized in the same way as those expressed positively. We can observe that, in all cases, the students of the experimental group had a better perception of the usability of the experiment under a cloud services scenario.

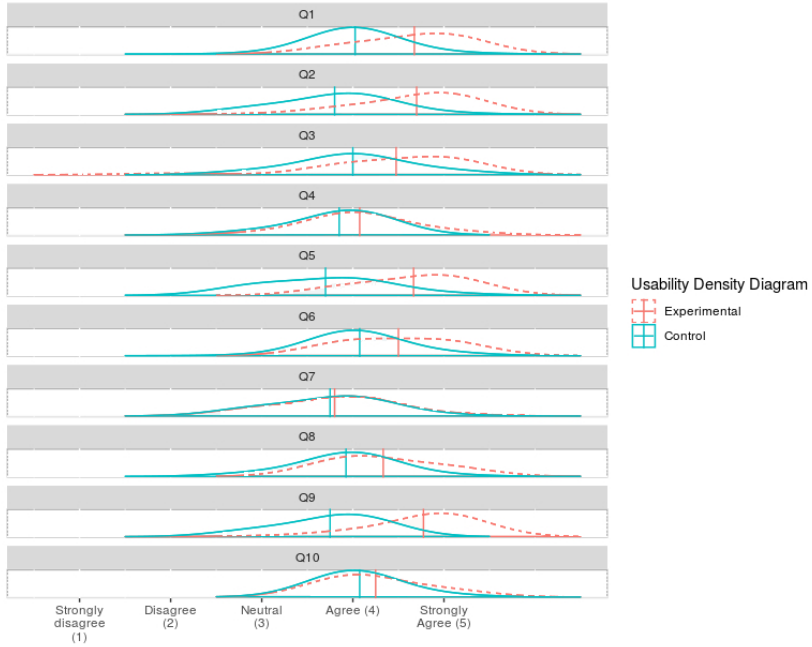


Figure 1 Diagram of the density of responses to the usability questionnaire for both groups.

In the case of the cognitive load questionnaire, in general, in each of the evaluation criteria, the results of the experimental group were closer to agreeing in comparison with the same results of the control group (Figure 2). In the case of the awareness indicator (questions 11–14), based on the responses of each of the groups, the experimental group showed better performance to find solutions and solve the experiment, compared to the control group. Regarding the indicator of commitment (questions 18–20), the experimental group showed a better perception of the relevance of the different types of learning activities concerning the needs and objectives of the students, which improves the cognitive load – satisfaction for these students.

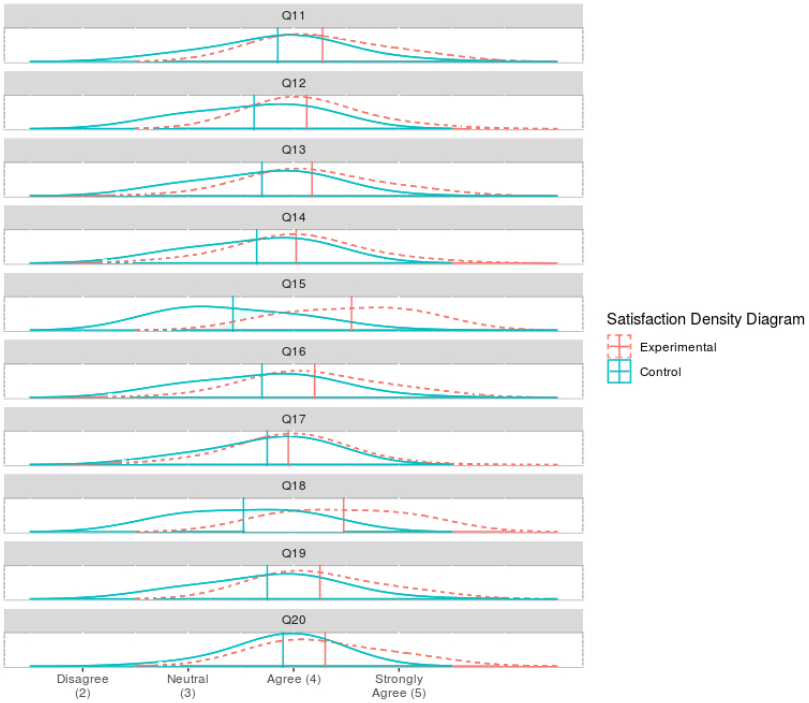


Figure 2 Diagram of the density of responses to the satisfaction questionnaire for both groups.

Regarding the challenge indicator (questions 15-17), the experimental group showed a better relationship between the degree of student satisfaction and the level of challenge they face; In this case, satisfaction has been the main reward of the students, which facilitates the additional memory load required, thus relieving the cognitive load and increasing the overall well-being of the students.

Making a comparison of the scores obtained for both the usability scale and the cognitive load scale, a scatter diagram was shown in Figure 3. In this diagram, we can see that the control group obtained a mean score slightly higher than 70 points in the case of the usability scale, equivalent to a rating of "Acceptable / Good" and slightly less than 70 points in the case of cognitive load. For the control group, the average of values obtained on the SUS scale is around 85 points, with a rating of "Excellent" and an average of 80 points for the case of cognitive load.

Taking into account the above, we can conclude that the students of the experimental group that uses instances of cloud services had significant benefits in terms of indicators related to usability and cognitive load compared to the control group of students who used the local database installation.

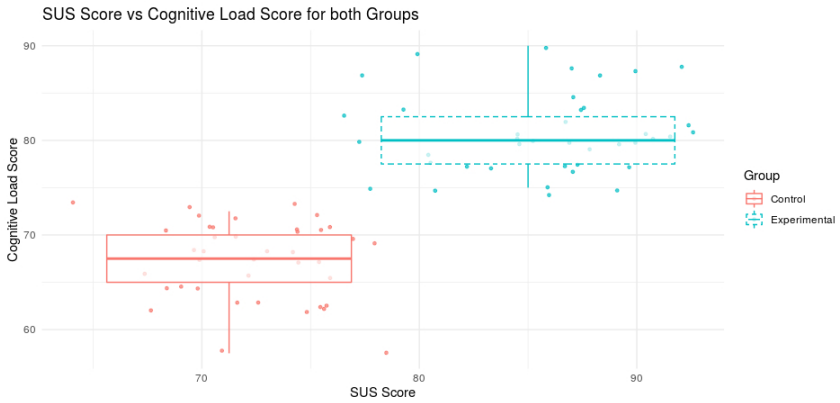


Figure 3 Comparison of scores for both groups.

Based on the above results, we can provide reliable answers to the research question established at the beginning of this document. For usability, both groups consider the development of the experiment as a pleasant experience, although the experimental group expresses greater satisfaction than the control group. In the case of the cognitive load, we can conclude that the experimental group has better results for the indicators of awareness, challenge, and commitment than the control group. For each of the questions related to these indicators, the control group students describe their experience as less satisfactory concerning the experimental group that used the cloud-based instance of the database. The satisfaction of the students in the experimental group has, in turn, a positive impact on the cognitive load and, subsequently, on their learning. Concerning the relationship of the cognitive load and usability scales, we see that the students of the experimental group have a better relationship between both scales than the control group, which shows that our model is adequate in determining that satisfactory learning is the result of a good relationship between usability and factors that tend to reduce cognitive load.

6 Conclusions and future work

In this work, we have shown the benefits that cloud services can present for students and education. A satisfactory learning model based on usability and cognitive load has been presented. It has been shown that cognitive load indicators have a positive impact on satisfactory learning in those students who use a cloud instance of data-base servers. On the other hand, we highlight how cloud services implementation for data-base servers shows better usability than one in which the servers are used locally. The main disadvantages that we find in a scheme based on cloud services are the dependence on a good internet connection and the need for a credit card to generate the free Google Cloud account. This is an ongoing work, which must be completed with another type of analysis, where the impact of the proposed model on academic performance is shown, as well as other factors that can benefit learning and complete its validation. Future work is aimed at demonstrating the validity of the model presented in more learning scenarios, with the final objective of providing a comparative study of all possible benefits, advantages, and limitations.

References

1. Fekry Fouad Ahmed. Comparative analysis for cloud based e-learning. *Procedia Computer Science*, 65:368–376, 2015.
2. Charles Bell. *Introducing the MySQL 8 Document Store*. Springer, 2018.
3. George R Bradford. A relationship study of student satisfaction with learning online and cognitive load: Initial results. *The Internet and Higher Education*, 14(4):217–226, 2011.
4. John Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.
5. Nicholas Carr. It in 2018: From turing's machine to the computing cloud. *An internet. com IT Management eBook*, 2008.
6. Tianping Dong, Yan Ma, and Lunpeng Liu. The application of cloud computing in universities' education information resources management. In *Information Engineering and Applications*, pages 938–945. Springer, 2012.
7. Gustavo Gutiérrez-Carreón, Thanasis Daradoumis, and Josep Jorba. Integrating learning services in the cloud: An approach that benefits both systems and learning. *Educational Technology & Society*, 18(1):145–157, 2015.
8. Nina Hollender, Cristian Hofmann, Michael Deneke, and Bernhard Schmitz. Integrating cognitive load theory and concepts of human–computer interaction. *Computers in human behavior*, 26(6):1278–1288, 2010.
9. Oksana Markova, Serhiy O Semerikov, Andrii M Striuk, Hanna M Shalatska, Pavlo P Nechypurenko, and Vitaliy V Tron. Implementation of cloud service models in training of future information technology specialists. In *Proceedings of the 6th Workshop on Cloud Technologies in Education (CTE 2018), Kryvyi Rih, Ukraine, December 21, 2018*, number 2433 in CEUR Workshop Proceedings, pages 499–515, 2019.
10. Srinual Nalintippayawong, Kanokwan Atchariyachanvanich, and Thanakrit Julavanich. Dblearn: Adaptive e-learning for practical database course: An integrated architecture approach. In *2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 109–114. IEEE, 2017.
11. Navin Sabharwal and Shakuntala Gupta Edward. Cloud sql. In *Hands On Google Cloud SQL and Cloud Spanner*, pages 27–55. Springer, 2020.
12. John Sweller. Cognitive load theory and e-learning. In *AIED*, volume 6738 of *Lecture Notes in Computer Science*, pages 5–6. Springer, 2011.

Using Code Review at School and at the Programming Club

Zuzana Kubincová[1]
Comenius University in Bratislava, Slovakia
http://sluzby.fmph.uniba.sk/ludia/en/kubincova1
kubincova@fmph.uniba.sk

Iveta Demková
Comenius University in Bratislava, Slovakia
Leasure-time Center, Šala, Slovakia
ivetkacs@gmail.com

Abstract

Educational code review is an activity that not only helps prepare future programmers into practice, but also teaches students to work with code in a different way. In the educational settings, activities focused on code review are mainly encountered at universities. In our research, we focused on lower levels of education and in our previous publications we presented the results of using code review at secondary school. Experimentally, we also tried to use the code review activities on the sessions at the leisure-time activities club. This paper provides a description of the research carried out. We compare the outcomes from the club with the results from the secondary school. We also give an overview of the benefits, as well as the problems such activities can bring to the classroom.

2012 ACM Subject Classification Human-centered computing → Empirical studies in collaborative and social computing; Applied computing → Collaborative learning; Social and professional topics → Computing education; Social and professional topics → Student assessment; Social and professional topics → Computing education programs

Keywords and phrases Code Review, Programming, Leisure-Time Activity Club, High School

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.14

Funding This work was supported from Slovak national project VEGA 1/0797/18.

1 Introduction

In school practice, many methodologies are used to teach programming. In our research we focused on code review and the possibilities of its use as an educational activity.

Code review is a technique commonly used in progra mming practice in software development. It can be characterized as examining the program code [1] and com-menting on it by other programmers, usually by colleagues of the program author. The aim of this activity is to find and correct errors in the program or to optimize the program code and thus improve its quality [5]. As several studies have shown, code review can help detect up to 70% of errors in software projects [9, 10]. Therefore, this technique is considered one of the best practices used by professional programmers in software companies. E.g. Google can serve as a good example of a software giant where code review has become an essential part of software development already since the beginning of the company's history [6].

In order to use code review in teaching, this technique needed to be modified. Thus, the so-called educational code review emerged, which many educational professionals have been dealing with recently in their research. When analyzing publications in this area, we found

[1] Corresponding author

several papers describing the application of code review into programming teaching, however, only at the university level [8, 7, 5]. As our target group is pupils at lower levels of education, we focused on them in our research.

2 Code Review in Our Programming Lessons

Since the beginning of the school year 2017/2018 we have been using the code review technique adapted to the school environment at high school, as well as at the leisure club, attended by pupils of different grades of elementary school, students of high school, and several adults as well. Code review conducted in the programming practice is a fairly complex process having its rules and standards. In introducing this activity into teaching, our goal was not to carry it out in the same way as it is common in development of software projects in programmer teams. We try to apply only some of its elements in teaching, to help learners develop their programming skills.

In some of our previous publications [3, 4], we already analyzed the results of the research, which was carried out at a bilingual five-year high school in 2017/2018. The first phase of the research was conducted in the third grade. Later, the next phase of the research was carried out in 2018/2019. The sample comprised some students who participated in the first phase, however, they were now in their fourth year of study.

In the first half of the year, 52 third-grade students participated in the research, whereas the second half had 55 participants of the third-grade. Teaching was carried out following the national curriculum. During the programming classes in the previous school years, the students used to program in Pascal programming language first and afterward they started to learn Python. At the end of the third grade, they were able to work with the timer and create their own mini-games. In the fourth grade, 10 students, participants of the informatics seminar, took part in the research. They mas-tered functions, lists, strings, text files, and various algorithmic calculations.

We introduced code review to the classroom teaching in two ways: using small re-views and project reviews. The small reviews were to review short programs prepared by the teacher. The students were assigned a short test on paper, consisting of two tasks. To solve the first task they had to find out what the first program would do if it was executed on a computer, to describe and possibly draw its output. In the other task, they had got a short program containing several errors. In the description of the program, there was written what this program would do if there were no bugs. The student's task was to find the errors, mark them and correct them.

In the other assignment type – the project review – the students reviewed longer programs created by their classmates and mutually commented on them.

At the same time, we conducted an experiment at a programming club, in the lei-sure-time center. Here we employed the first type of code review activity – small reviews. Five people of different ages and with different previous programming experiences participated in the programming club: a pupil of the fourth grade and a pupil of the eighth grade at elementary school, both having previous experience only with programming in Scratch; a freshman at a high school who simultaneously programmed in Python language in informatics classes at school; and two adults who had only a very basic experience in programming in Basic language from their high school years long time ago.

The club sessions were held once a week and lasted for 90 minutes. Our aim was to teach participants the basics of programming in Python. Gradually, the following topics were taught: variables, basic graphical commands (tkinter library), random numbers, for-loop,

functions, mouse click and keyboard events, conditions, text strings, timer, canvas object movement. Thanks to the higher time subsidies at the club it was possible to familiarize the club members also with such programming concepts and topics, which were not taught at the compulsory informatics classes at the high school and only were discussed at an optional informatics seminar.

The code review was involved in the activities of the club after explaining and practicing each topic. Participants reviewed two given programs within 10 minutes (5 minutes each). As with the small reviews at high school, one of the tasks was to find out what the program would do and the other one to look for errors in the given code. Although the regular club lessons proceeded in a looser style and the participants could cooperate and consult each other, this was not the case during the reviews. This activity was perceived as a "test without any grade", so everyone had to deal with it by themselves.

3 Programming Club Results

Every small review referred to the last topic taught. The first review consisted of tasks dealing with the for-loop, the second review covered the functions, the third one addressed the conditions, the fourth one the timer, and the fifth one moving the canvas objects.

The following figures depict how small reviews looked like – there is an example of one type of task in Fig. 1 and the other type of task in Fig. 2.

Club participants were evaluated by points for solving these tasks. A maximum of 5 points could have been earned for each code review. For the first task, in which they were to write what the program would do, they could get 2 points, in the second task, where they were supposed to find and correct errors, they could get 3 points.

The results of the small reviews at the programming club are shown in Table 1. Most reviews were made by only four out of five participants, as one of them did not attend the club meetings regularly.

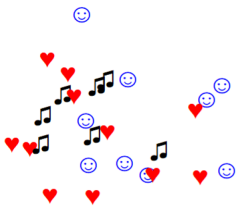
Find out what this program would do if it was executed. Describe it verbally and draw a picture as well.

```
import tkinter
from random import *
c=tkinter.Canvas(width=600,height=600,bg="white")
c.pack()

def drawing():
    x = 30
    y = 50
    v = 30
    for i in range(0,6):
        if i%3==0:
            c.create_rectangle(x,y,x+v,y+v,fill="blue")
        elif i%3==1:
            c.create_rectangle(x,y,x+v,y+v,fill="red")
        else:
            c.create_rectangle(x,y,x+v,y+v,fill="yellow")
        x = x + v
drawing()
```

Figure 1 Example of a code review task (type 1).

The following program should draw something similar to the picture on the right. However, there are several errors in the program. Find and correct them all.



```
import tkinter
from random import *
c=tkinter.Canvas(width=600,height=600,bg="white")
c.pack()

def tap(event):
    x = range(50,550)
    y = range(50,550)
    if event.keysym = "h"
        c.create_text(x,y,text="♥",font="Arial 30",fill="red")
    elif event.keysym = "s"
        c.create_text(x,y,text="☺",font="Arial 30", fill="blue")
    else
        c.create_text(x,y,text="♫",font="Arial 30",fill="black")

c.bind_all("<Key>", tap)
```

Figure 2 Example of a code review task (type 2).

In the first review, none of the club participants earned any point. No one was able to figure out what would be the result of the first program or find and correct the errors in the second program. The problem was probably in completely new types of tasks they had never encountered before.

The situation changed significantly in the second review when two participants even scored full points and the remaining two earned four and three points. Participants also achieved very similar results in all other rounds of code review. The best results were achieved in the last round – two participants scored full points and the other two lost only one point.

The attitude of the club participants to this activity was explored based on the observation of their work at the club sessions and on the personal interview with them. Their reactions during the interview were positive. While watching their work, we noticed that they often were able to work out both tasks in less time than was available.

Table 1 Code review outcomes at the programming club.

Participant	Gender	Code Review 1	Code Review 2	Code Review 3	Code Review 4	Code Review 5
4th-grader	**male**	0	5	4	3	4
adult	**male**	0	5	5	5	5
8th-grader	**male**	0	4	5	4	5
adult	**female**	0	3	1	5	4
high school student	**male**	-	-	4	-	-

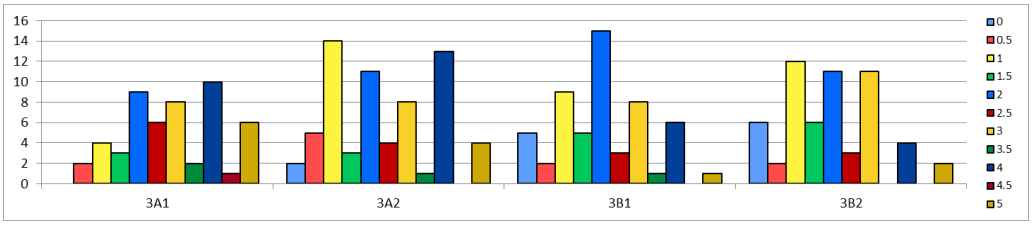


Figure 3 The raw score for all review rounds by groups.

4 Comparison with the Results at High School

Our previous research at high school was conducted in two phases. In the first phase, the research sample consisted of students of two classes in the third grade, divided into four groups (A1, A2, B1, and B2). Although the code review activities involved small reviews as well as project reviews, since we want to compare the results with the results attained at the programming club, we will only present the results of high school students from small reviews. These were carried out at high school and later on in the club in the same way – in five rounds. The programs reviewed by the students and club participants were also the same.

The best results in this activity in high school were achieved by the A1 group, in which there were no reviews with zero rating and three students scored full score in two code review rounds. The A2 group took second place. One student in this group got the full score in two small reviews and two other students in one review. Only one student received a zero rating in two review rounds.

Class B was less successful than class A. In both of its groups, there were 4 students with zero score in one small review and in the B2 group another student has got zero rating from two reviews. Only one (B1) and two (B2) students earned the full score, however, from just one review.

The summary results for all rounds of code review by groups are depicted in Fig. 3. The chart shows the number of students in each group who have earned the appropriate score throughout all review rounds.

After the last review round, students admitted that while these "small tests" were difficult for them, they helped them learn to look for errors in the program and think differently about the program code. So far in the informatics classes, they have only encountered the approach: "I have a problem, my task is to code the program to solve it". During the code-reviewing, they came to the opposite side: "I have a program code, my task is to find out what the program is doing".

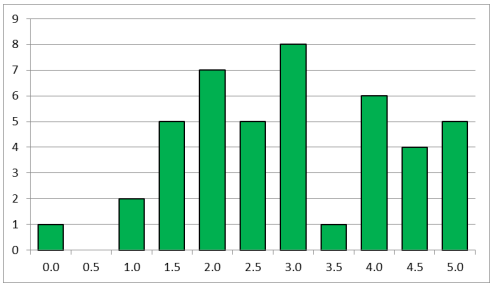


Figure 4 The raw score of fourth-grade students.

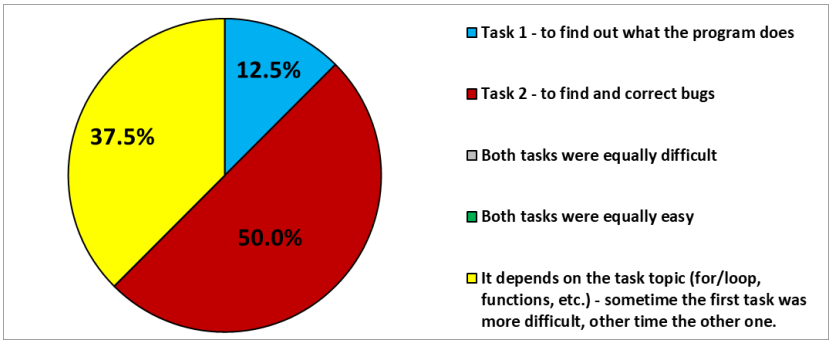


Figure 5 Answers to the question "Which of these two task types did you find more difficult to solve?".

In the following school year (phase two), we tried to verify the previous research results with fourth-grade students who passed small reviews in their third grade. They got programs for small reviews that were identical to those from the previous year. However, the results did not show any significant facts to suggest that students improved in finding errors and finding out what the program would do if it was executed. The raw score of fourth-grade students is shown in Fig. 4.

The detailed results from the third and fourth grades can be found in the thesis of one of the authors [2].

An analysis of the results of the small reviews in both the third and fourth grades shows that the first task type in which students were supposed to find out what the program would do came generally off better than the other one. This finding was confirmed by the fourth-grade students in their responses (Fig. 5) to the questionnaire administered after all the code review activities were completed.

Their opinion on the use of such types of activities in programming teaching was also surveyed by the question, whether they would include such types of tasks in the programming teaching in other grades or schools (Fig. 6). Most students responded positively, saying that this is good training in learning to code. The remaining students did not know how to comment on the question; no one answered negatively.

It is not quite straightforward to compare the results obtained in the school environment with those of the leisure-time club. The programming club is mainly attend-ed by people who are really interested in coding and are therefore showing better results. On the other hand, informatics classes and programming at high school are mandatory subjects, no matter

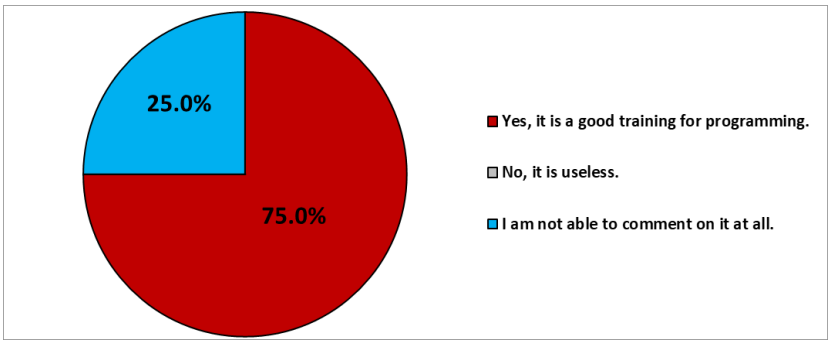


Figure 6 Answers to the question "Would you incorporate the tasks like this into programming teaching also in other grades/schools?".

what is the relationship of students to these subjects. In addition, in most schools, the informatics is taught once a week one teaching hour (i.e. 45 minutes). A new topic is often presented in one or two lessons. In the club, each new topic was also explained and trained during one or two club sessions, each session lasting 90 minutes. During this time, the participants have solved considerably more tasks than the students in the high school research did. Another difference is in the size of the group taught by the teacher at school or by the instructor in the club. The leisure-time clubs are usually attended by a much smaller number of participants than the number of students in the classroom, making it easier for the instructor to implement the individual approach method.

The results of experimental research at the programmer's club not only confirmed the anticipated findings that its participants, who were interested in programming and disposed to learn programming, were willing to spend more effort and time to learn to code but also showed that such individuals can quickly accept also less traditional educational activities and show very good results in them. In our case, after their first experience with code review, they understood what was expected of them and improved their reviewing skills both in identifying what the program was doing and in detecting and fixing errors.

5 Educational Code Review Benefits and Problems

Implementation of code review in teaching is not entirely trouble-free and brings with certain complications. In the school environment, the lack of time can be a problem. Implementing code review activities in the form of small reviews requires regular reserving of part of the lesson, which may not be easy as the teacher is supposed to explain and exercise with students quite a lot of topics according to the national curriculum, while the time subsidy for informatics is 45 minutes a week. In addition, since this is a type of activity that students do not usually encounter at school, it is necessary to explain in detail at the beginning what they are expected to do, to show them sample solutions to similar assignments, or to train them for the reviewer role. Again, all of these activities require extra time. However, they are really necessary when reviewing larger programs, such as projects, and certainly also very helpful in the case of small reviews.

Considering the involvement of code review activities at the programmers' club, the time constraints are eliminated as the club has both a larger weekly time subsidy and a looser schedule of activities.

Code reviewing also brings certain benefits to programming teaching. In this way, the teacher gains a broader view of the student, learns how they are doing with programming, but also how they can read somebody else's code, understand it, find out what is the result of the program, find mistakes in it and correct them. The student can get an alternative view of solving problems often the same ones they have already solved themselves, by which they basically learn to program in another way, learn to understand other people's programs, correctly describe mistakes and also develop their communication skills. These and other benefits of using code review in teaching programming, whether at school or at a club, will be enhanced if such type of activity is used to review longer program codes, especially if the students themselves are the authors (see e.g. project reviewing presented in one of our previous publications [4]).

6 Conclusions

Programming practice, research from foreign universities, and also our research show that these activities can gradually improve programming skills. We observed improvement in the quality of the program code even in the case of beginners in programming. Therefore,

in spite of problems with the time subsidy of the subject, we think that teachers should incorporate similar activities into the teaching of programming so that the students gradually get used to them and as soon as possible, can use the benefits these activities offer them.

When employing such activities, it can be useful for the teacher to note during the lesson what mistakes are made by the students in creating programs, what error mes-sages they often encounter, etc. Based on these observations, the teacher can incorporate new elements into programming teaching, for example, also through the code review. This way, it can also be found out whether the student can think about the program, understand it and visualize in abstraction what the program will do and describe it afterward.

It is advisable if activities of this type are carried out with students regularly and especially in the long term. This, too, can help them get deeper into programming and build proper programming habits right from the start.

Our research is currently continuing at the programmer's clubs in the leisure-time center. Several participants from the last year's club continue to attend our club sessions devoted to programming in Python. There is also a new group of beginners learning to program in Python.

With the past and new members of the club, we continue to carry out code review activities. Our goal is to conduct the research on a larger sample so that we can verify previous results and better understand the potential of educational code review in learning to program at leisure-time clubs and its portability to teaching at school.

References

1 Barry W. Boehm. Software engineering economics. *IEEE transactions on Software Engineering*, SE-10(1):4–21, 1984. doi:10.1109/TSE.1984.5010193.

2 Iveta Csicsolová. Code review v informatike na strednej škole. Master's thesis, Comenius University in Bratislava, Slovakia, 2019. Diploma Thesis, in Slovak.

3 Zuzana Kubincová and Iveta Csicsolová. Code review in high school programming. In *2018 17th International Conference on Information Technology Based Higher Education and Training (ITHET)*, pages 1–4. IEEE, 2018. doi:10.1109/ITHET.2018.8424617.

4 Zuzana Kubincová and Iveta Csicsolová. Code review at high school? yes! In *International Conference in Methodologies and intelligent Systems for Techhnology Enhanced Learning*, pages 115–123. Springer, 2019. doi:10.1007/978-3-030-23884-1_15.

5 Zuzana Kubincová and Martin Homola. Code review in computer science courses: Take one. In *International Conference on Web-Based Learning*, pages 125–135. Springer, 2017. doi:10.1007/978-3-319-66733-1_14.

6 Caitlin Sadowski, Emma Söderberg, Luke Church, Michal Sipko, and Alberto Bacchelli. Modern code review: a case study at google. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, pages 181–190, 2018. doi:10.1145/3183519.3183525.

7 Mason Tang. *Caesar: a social code review tool for programming education*. PhD thesis, Massachusetts Institute of Technology, 2011.

8 Deborah A. Trytten. A design for team peer code review. *ACM SIGCSE Bulletin*, 37(1):455–459, 2005. doi:10.1145/1047124.1047492.

9 Hidetake Uwano, Masahide Nakamura, Akito Monden, and Ken-ichi Matsumoto. Analyzing individual performance of source code review using reviewers' eye movement. In *Proceedings of the 2006 symposium on Eye tracking research & applications*, pages 133–140, 2006. doi:10.1145/1117309.1117357.

10 Karl Eugene Wiegers. *Peer reviews in software: A practical guide*. Addison-Wesley Boston, 2002.

Learning Resources with Augmented Reality

Lázaro V. O. Lima
Centro ALGORITMI, Universidade do Minho, Braga, Portugal
lazaro.lima@ifb.edu.br

Cristiana Araújo
Centro ALGORITMI, Universidade do Minho, Braga, Portugal
decristianaaraujo@hotmail.com

Luis Gonzaga Magalhães
Centro ALGORITMI, Universidade do Minho, Braga, Portugal
lmagalhaes@dsi.uminho.pt

Pedro R. Henriques
Centro ALGORITMI, Universidade do Minho, Braga, Portugal
prh@di.uminho.pt

Abstract

Preparing teachers and students for a connected and programmed world depends on how we develop and reinvent teaching tools. The society has realized and is absorbing Computational Thinking and its related skills. The pragmatics shows that a person only acquires a new way of thinking or a new way of behaving if he is trained with the appropriate devices. Computational Thinking should be training from an early age to acquire important skills; in that way, the interpretation and design of algorithms/programs will become much easier. However, the development of Computational Thinking requires the creation and use of appropriate Learning Resources (LR). We will discuss how an ontology can be used to specify what is involved in Computer Programming and how these concepts and Computational Thinking concepts are related. We believe that this formal description will guide the choice of convenient LR. In that context, we intend to investigate the impact of Augmented Reality on them. After presenting the ontological approach, the paper will focus on the process of shaping Computational Thinking through Augmented Reality. We aim at creating AR-based LR prototypes to validate the idea we present here. We are convinced that an attractive way to improve fundamental skills is necessary to practice and use these tools with young students, but LRs must be attractive, motivating and effective.

2012 ACM Subject Classification Computing methodologies → Mixed / augmented reality

Keywords and phrases Computational Thinking, Learning Resource, Augmented Reality, Teacher Support Tools

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.15

Funding This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020.

1 Introduction

To promote the development of fundamental skills in the students like reading, writing, arithmetic, or analytic capabilities, nowadays we need to adopt new strategies to teaching and learning process.

These skills are crucial for many activities that in general require Problem-Solving capabilities as it is the case of Computer-based tasks demanding for Computational Thinking (CT) ability. In that direction, CT shall be included as a fundamental skill in the school curricula. The aim of such a decision is to develop in the student competencies for problem-solving that will be required to the 21st-century citizens. To train and induce CT in the student a novel teaching/learning process must be devised using techniques derived from

Mathematics, Gaming, and Computer Science. The abilities that characterize CT – like logic reasoning, abstraction, rigor in analysis and specification, strategic planning, etc. – are of uttermost relevance in Programming. Augmented Reality(AR) can provides greater motivation, gains learning, and delights the students who use it. AR is defined by Azuma [3]as the overlapping of virtual information in the real world through technology. This information can be simple textual images or 3D objects. Unlike Virtual Reality where the user is fully immersed in the environment and visual sense is controlled by the system, AR increases information in the real world, the user maintains a sense of presence in the real world and requires mechanisms to combine the real world with the virtual one. Unlike Virtual Reality consists in the simulation of virtual scenes, raising the user to an experience of immersion and interaction in a virtual world based on simulation generated by computer. AR supports pedagogical approaches through constructivism learning by enabling educational experiments that complement the activities of the real classroom like [24], one of the works that explore AR as a pedagogical tool. We believe that it is possible to explore AR as a technology that provides constructs to develop skills of uttermost importance for computer programming, not only as a mere technology operator, but also as a computationally literate individual. AR in education can be applied in the training of students' abilities, encouraging learning based on discoveries. Summing up, AR can be used to provide a rich contextual learning environment, adhering to constructivist principles, fostering opportunities for multiple learning styles, engaging learners in ways that are not possible in real-world without real consequences if mistakes are made during the training. These advantages will be used to created LR that should be available to promote CT in schools. Our goal is to show that we can mix technological facilities for the creation of new tools such as Augmented Reality.

This paper is organized in five sections, Section 2, with objectives and Research Methodology, Section 3 Computational Thinking is described, in Section 4, *OntoCnE*, a Ontology to describes the Computational Thinking domain, in Section 5, the work in progress and Section 6 for conclusions paper.

2 Objectives and Research Methodology

The information in the AR is apprised in real-time that provides an increase in the attention of the students [15]. Thus the AR can be a powerful allied technology in the development of CT in students exploring in different ways the decomposition, pattern recognition, abstraction and algorithm construction[7]. With the popularization of games and applications that use AR, there is also the adaptation for different platforms, including mobile platforms, in which the use of mobile phones to aid in education using AR is possible today.

We will use OntoCnE ontology to describe CT. This ontology is discussed with more details in the works of Araújo [1]. Consequently, the main objective of the research is improve motivation in teaching CT, creating atool that with formal descriptions of an Ontology-driven Learning Resource designed to describe CT will result in LR using Augmented Reality techniques.

The methodology used in this research to achieve our objectives will be Design Science Research, this methodology focuses on the development and performance of artifacts with explicit intentions of functional improvement of the developed artifact. DSR is most commonly applied in the development of artifacts such as algorithms, interfaces, methodologies design, languages. DSR's focus is to develop the knowledge to design solutions to problems in a particular field according to [20]. The DSR consists of a sequence of activities that produces an innovative product, with the artifact created it is possible for the researcher to better

understand the problem and have a better view to reassess the problem and thus improve the quality of the design process in a construction loop until developing a final artifact to solve complex and relevant field problems. Because the focus of this methodology is on creating an artifact for a given problem, this artifact must be well evaluated to ensure its objectives. It also should solve a problem that has not been resolved or provide a better solution. For developing this artifact we use the DSR methodology, Hevner [10] counts with seven guidelines to follow: Problem identification and Motivation; Objectives of a Solution (research to define objectives); Design and Development (creation of the artifact); Demonstration (used in appropriate environment); Evaluation (performance of the artifact); Communication.

3 The Importance of Teaching Computational Thinking

In 2006, Wing [21, 22, 23] proposed the foundations of CT and showed how society is influenced by technology even more in education. According to the author, the most important and high level thinking process is the process of abstraction, being used in the definition of patterns, generalizing from specific instances and parametrization. CT is a method for solving problems, or designing systems and understanding human behavior, based on the fundamental concepts of computer science, that develop competencies in students required in the 21st century.

Computational Thinking is based on the concepts of Pattern Recognition, Abstraction, Problem Decomposition, Algorithms. Moreover students acquiring those skills to solve problems, are also able to debug and assess the calibration of the proposed solution. In terms of the use of technological resources, in order to introduce the concepts of programming languages, it is important to pay attention that they must be able to motivate and encourage the students, development of abstraction, decomposition of problems and the organization of steps to solve a problem. Moreover they should allow for a constructivist-based teaching approach, that has been proved to promote effective knowledge acquisition. The ability to formulate algorithms for computers is like building instructions for a computer to solve / repeat processes; this action is related to solving simple or complex tasks, but learning how to build algorithms has a great cognitive load and needs to be trained since young as explained in the work done in an effort to incorporate CT in curricula[16].

With adequate resources it is possible to work with the identification of common characteristics between the problems and their solutions. We can further identify patterns among the sub-problems that have been abstracted, finding an efficient solution to the problems encountered. It is also possible to work with resources that help breakdown processes in smaller parts for easier resolution. A learning activity can use a LR as an unplugged activity or a game as demonstrated in work of [11]. Thus it is possible to prepare the thought so that it arrives at the moment of creation of the Algorithms in the strategy or clear instructions for the solution of the problem.

Resnick [18] explores CT, but the use of AR can be observed in the works of [17, 12]. The exploration of [12] takes the earlier work of *CodyRoby* into a low-cost AR mobile system, which uses a simple smartphone as an augmented sensor to transform a fully disconnected coding set into an Augmented Reality coding experiment. The relationship between the development of CT and programming learning with AR can also be seen currently in the investigation of [13, 19, 8].

4 Describing Computational Thinking with an Ontology

An Ontology, in Computer Science, represents a set of concepts within a domain and the relationships between them. *An ontology is an explicit specification of a conceptualization* [9]. Furthermore used to represent knowledge and to perform inference on domain objects. We felt the need for a formal definition of the domain we are coping with. In that sense we decided to describe it creating a specific ontology that describes the domain of CT in [1].

The ontology we create, which is called *OntoCnE* [1], describes the CT domain, more specifically how to teach it and what material is needed to teach it in the various years of schooling. After the construction of the ontology, we select the concepts that would be taught in each school year and add new concepts to ontology. This ontology will allow to classify the resources that will be used to train a certain concept at a given level of education. The research proposed by Azevedo [2] describes Micas, as a tool that allows to store the resources and classifies them according to the *OntoCnE*, the tool can be accessed at `https://micas.epl.di.uminho.pt/`. After getting to know a part of *OntoCnE*, in the next section we will present how the working tool will be developed. In the following fragment, we can see concepts taught in the 1st year of *OntoCnE*.

Listing 1 Concepts taught in the 1st year (fragment).

```
Triplos{
ano1 =[
    desenvolve=> PensamentoComputacional,
    desenvolve=> RaciocinioLogico,
    desenvolve=> Abstraccao,
    apresenta=> Problema,
    introduz=> Algoritmo,
    introduz=> Instrucao,
    introduz=> Programa,
    introduz=> DispositivoDigital,
    introduz=> LingGrafica,
    usa=> Computador,
    usa=> Robot ];
}
```

5 Learning Resources with Augmented Reality

Learning Resources is a tool that helps teachers in teaching and student on learning. LR are hard or soft devices that allow students to train previous knowledge or acquire new knowledge, stimulating their ability to comprehend, organize and synthesize educational content in a specific domain [4]. The LR can be simple and developed based on drawn letters or printed at home, demonstrating simplicity and accessibility for use. There are activities directly linked to programming logic as activities related to loops, sequences, events, conditionals, working with binary numbers, or even activities directly linked to the training of CT.

[1] From the Portuguese *Ontology for Computation in School*

There are *Code.org* [2], activities like *CT with Monsters* in [14] can be adapted, task focused on decomposition, then students will analyze a catalog of monsters for patterns, abstract similar details from the monsters, then use that information to create an algorithm (instructions) for other students to draw a certain monster. Students can alternate algorithms with another group and test to see each others result (Debug).

It is crucial to have adequate resources to train the different skills involved in CT. The more resourceful, motivating and effective, the better students will shape their minds by learning the skills they desire.

Our goal with this paper is to show that we can mix a technological facility, AR, to build Learning tools that as the literature describes, LR with AR will increase student motivation as [6, 5] shows in his work, applying AR resource as a teaching tool not only can create a learning environment. Bearing in mind that the generated AR-LR should work in the Web Browser or on other platforms that do not require high computing power neither and complex, expensive operating equipment to be purchased by schools.

With a work environment formed with a generated AR-LR, having as one of its goals, represent analogies to understand complicated programming concepts. This definition, ruleset, and operation step using *OntoCnE* will allow us to quickly generate and modify the new AR-LR.

In the last step, we will study how to start generating a description to later generate new artifacts, with functionalities integrated with real activities and alternative technologies. These artifacts will provide the path to create the appropriate LR according to the description of Ontology for CT. Thus it is possible for students to understand the general levels of programming and how to think algorithmically to arrive at a solution. Hence understanding the concepts necessary for CT. The construction of such a tool will also focus on usability according to the studies reported in previous section. The tasks performed by students in LR activities can not be difficult or too easy to avoid disinterest in students.

A first proposal of the system architecture is depicted in Figure 1, where can be seen that the main users of the system will be the teacher and the resources development. The teacher using an appropriate tool will have an impruve in his learning activities. The methodology adopted prescribe the execution of a sequence of activities that produces an innovative product. With the created artifact, it is possible for the User and the Researcher to better understand the problem and have a better view to reassess the problem and, thus, improve the quality of the design process in construction.

The artifacts to be developed aiming to impact on the development of Computational Thinking using Augmented Reality, shall assure that the interaction with the virtual objects will be during the use of the tool, not just showing a simple 3D object to the student. The participation of the teachers or future users of the system in the process of construction of the functional requirements is primordial. The description of CT through the ontological approach, will be carried out by *OntoCnE*. Libraries that allow the use of AR, integrate the system in order to be included into the 3D Object behavior rules.

Then exemplifying the ideas, in figure 2 we can observe the possible interactions, the student changes the properties of 3D objects and visualizes 3D geometric shapes in AR. Thus, we have the introduction of concepts of CT and programming using Spatial Geometry with AR, in which students have difficulty in having the abstract notion of geometric figures.

First using *OntoCnE* to describe the CT. The architecture will be defined to develop AR artifacts. Micas be used to create an LR repository to make it available to teachers. *OntoCnE* describes the domain of CT and the objective is to assist in the classification

[2] `https://code.org/curriculum/unplugged`

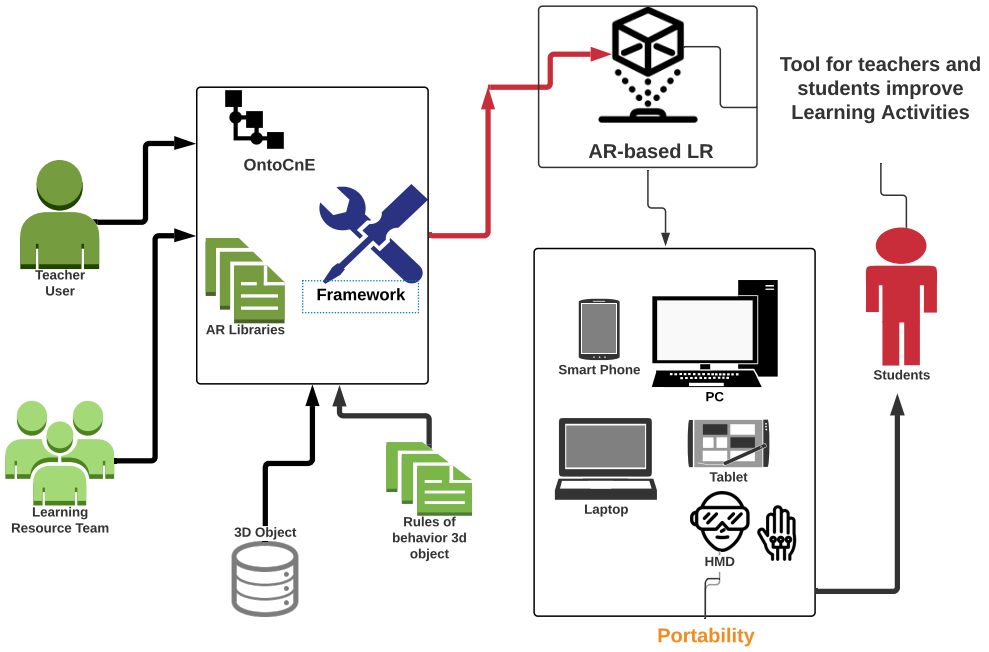


Figure 1 Proposed Scheme to generate AR-LR.

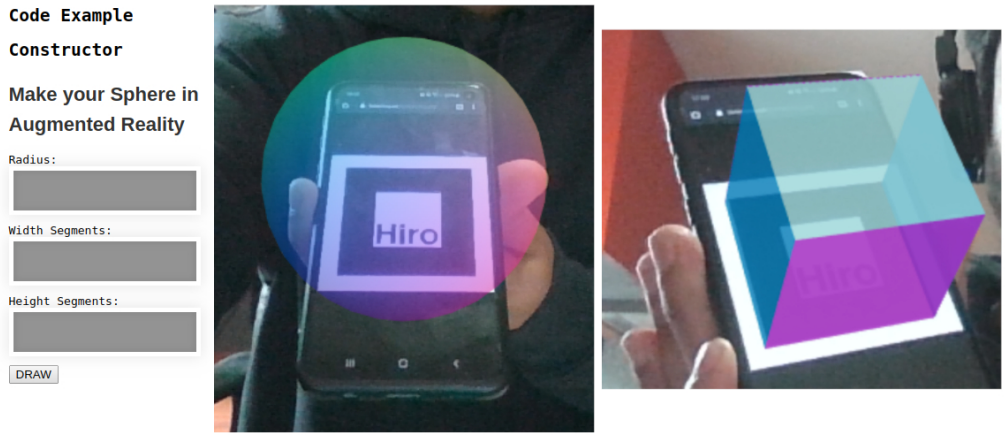


Figure 2 Example of the interaction in the prototype under construction.

of LR. It is important to highlight that we want to demonstrate that Augmented Reality contributes to the development of skills related to Computational Thinking, so Plugged Learning Resources will be generated. We see that the main users of the system will be the teacher and the LR development. The tasks performed by students in LR activities can not be difficult or too easy to avoid disinterest in students. After performing the activities related to the objectives of the artifact, a final survey will be verified, the improvement in the skills or not through the analysis of the results.

6 Conclusion

To learn Computer Programming is necessary to analyze a problem, to design solutions, test and optimize solutions, not only coding. Computational Thinking helps expressing how to solve a problem. CT involves concepts like decomposition, abstraction and algorithmic design. With those ingredients, CT potentiates the ability to program computers. Learning Resources are crucial to train properly CT, so the more complete and wiser they are, the more effective their help. We believe that smart choices must be made to create adequate LRs. In that context, we research the inclusion of Augmented Reality components in some devices to create new, improved, LRs.Producing those Augmented LR is challenging and an interesting task but time consuming if done manually. So in this paper we suggested the use of a generation mechanism capable of producing the required resources. The process will be guided by *OntoCnE*, an ontology for CT, to provide a formal representation of the knowledge domain. That automation platform will leverage the production and availability of the adequate effective resources. However, in order to evaluate the impact of Augmented LRs on CT, we will design and conduct experiments with real students in real classrooms to measure the results in learning activities with and without AR. AR technology can provide animation, sound, and video to make traditional resources more appealing, and more helpful transmitting information.. Guided by OntoCnE, the prototype will have the combination of introduction to programming, presenting definitions and properties, and teaching spatial geometry in mathematics. Using Augmented Reality to interact with the created 3D objects.

The ongoing work is devoted to build prototypes to validate the idea presented here. To get the most out of CT skills, we are convinced that it is necessary to practice and use well chosen Learning Resources, but LRs must be attractive, motivating and effective.

References

1 Cristiana Araújo, Lázaro Lima, and Pedro Rangel Henriques. An Ontology based approach to teach Computational Thinking. In Célio Gonçalo Marques, Isabel Pereira, and Diana Pérez, editors, *21st International Symposium on Computers in Education (SIIE)*, pages 1–6. IEEE Xplore, November 2019. doi:10.1109/SIIE48397.2019.8970131.

2 Ana Azevedo, Cristiana Araújo, and Pedro Rangel Henriques. Micas, a Web Platform to Support Teachers of Computing at School. In A. J. Osório, M. J. Gomes, and A. L. Valente, editors, *Challenges 2019: Desafios da Inteligência Artificial, Artificial Intelligence Challenges*, pages 625–641. Universidade do Minho. Centro de Competência, 2019-05.

3 Ronald T. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, 1997. doi:10.1162/pres.1997.6.4.355.

4 Rona Bušljeta. Effective use of teaching and learning resources. *Czech-Polish Historical and Pedagogical Journal*, 5(2):55–70, 2013.

5 Su Cai, Enrui Liu, Yang Yang, and Jyh-Chong Liang. Tablet-based ar technology: Impacts on students' conceptions and approaches to learning mathematics according to their self-efficacy. *British Journal of Educational Technology*, 50(1):248–263, 2019.

6 Wen-Hung Chao and Rong-Chi Chang. Using augmented reality to enhance and engage students in learning mathematics. *Advances in Social Sciences Research Journal*, 5(12), 2018.

7 Cheng-Yu Chung and I-Han Hsiao. An exploratory study of augmented embodiment for computational thinking. In *Proceedings of the 24th International Conference on Intelligent User Interfaces: Companion*, pages 37–38. ACM, 2019.

8 Xiaozhou Deng, Qiao Jin, Danli Wang, and Fang Sun. Arcat: A tangible programming tool for dfs algorithm teaching. In *Proceedings of the 18th ACM International Conference on Interaction Design and Children*, pages 533–537. ACM, 2019.

9 Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In *International Journal of Human-Computer Studies*, pages 907–928. Kluwer Academic Publishers, 1993.

10 Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *Management Information Systems Quarterly*, 28(1):6, 2008.

11 Cagin Kazimoglu, Mary Kiernan, Liz Bacon, and Lachlan Mackinnon. A serious game for developing computational thinking and learning introductory computer programming. *Procedia-Social and Behavioral Sciences*, 47:1991–1999, 2012.

12 L Klopfenstein, Andriy Fedosyeyev, and Alessandro Bogliolo. Bringing an unplugged coding card game to augmented reality. *INTED Proceedings*, pages 9800–9805, 2017.

13 Divna Krpan, Saša Mladenović, and Biserka Ujević. Tangible programming with augmented reality. In *12th International Technology, Education and Development Conference*, 2018.

14 CODE ORG. Available in `https://code.org/`, 2015. Access date: set 2019.

15 Mark Petrorvich, Mamta Shah, and Aroutis Foster. Augmented Reality Experiences in Informal Education. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, pages 815–819, 2019. `doi:10.1109/tale.2018.8615396`.

16 Jake A Qualls and Linda B Sherrell. Why computational thinking should be integrated into the curriculum. *Journal of Computing Sciences in Colleges*, 25(5):66–71, 2010.

17 Iulian Radu and Blair MacIntyre. Augmented-reality scratch: a children's authoring environment for augmented-reality experiences. In *Proceedings of the 8th International Conference on Interaction Design and Children*, pages 210–213. ACM, 2009.

18 Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay S Silver, Brian Silverman, et al. Scratch: Programming for all. *Commun. Acm*, 52(11):60–67, 2009.

19 Chin-Hung Teng, Jr-Yi Chen, and Zhi-Hong Chen. Impact of augmented reality on programming language learning: Efficiency and perception. *Journal of Educational Computing Research*, 56(2):254–271, 2018.

20 Joan Ernst Van Aken. Management research as a design science: Articulating the research products of mode 2 knowledge production in management. *British journal of management*, 16(1):19–36, 2005.

21 Jeannette M Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.

22 Jeannette M. Wing. Computational thinking - What and why? *The Link Magazine*, 2011. URL: `http://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why`.

23 Jeannette M. Wing. Computational thinking benefits society. *40th Anniversary Blog of Social Issues in Computing*, 2014(3):33, 2014. `doi:10.1145/1118178.1118215`.

24 Danny Yaroslavski. How does lightbot teach programming. *Retrieved January*, 29:2016, 2014.

Computational Thinking Education Using Stickers and Scanners in Elementary School Classes

Akiyuki Minamide
International College of Technology, Kanazawa, Kanazawa-shi, Ishikawa, Japan
minamide@neptune.kanazawa-it.ac.jp

Kazuya Takemata
International College of Technology, Kanazawa, Kanazawa-shi, Ishikawa, Japan

Hirofumi Yamada
Kanazawa Institute of Technology, Hakusan-shi, Ishikawa, Japan

Abstract

Programming education will be compulsory at elementary schools from fiscal 2020 in Japan. Programming education in elementary school does not teach programming language coding, but computational thinking. This paper describes a new programming education method using stickers and a scanner that combine the features of unplugged programming and physical programming. The new materials developed in this study offer superior features compared to commercial materials, such as low cost, use in lower grades class in elementary school, and no need for teacher ICT skills. Demonstration experiments were conducted on 66 third-grade elementary school students to confirm the effectiveness of the materials. The children used the new teaching materials without being confused, and the teachers were able to smoothly teach. From this result, it was confirmed that this teaching material could be used in the lower grades class of elementary school.

2012 ACM Subject Classification Social and professional topics → Computer science education

Keywords and phrases computational thinking, programming education, programming with stickers

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.16

1 Introduction

In Japan, programming education will be compulsory at elementary schools starting in fiscal 2020. Programming education in elementary schools does not teach programming languages as higher education institutions do but teaches computational thinking [7, 8]. However, there are some problems with introducing programming education in elementary school.

Japanese elementary schools have 30 to 35 children per class, and one teacher must be in charge of one class. Although programming materials used by a small number of children are commercially available, there is no teaching material intended for large classes. In addition, elementary schools do not have sufficient budget for facilities such as ICT (Information and Communication Technology) devices and robots including personal computers, and elementary school teacher does not have programming skill and knowledge to teach children. In order to solve these problems, it is necessary to consider programming education throughout the school and society, and new teaching materials that require less capital investment and have nothing to do with the programming skills of teachers are needed.

In this paper, we describe a new programming education method using stickers and a scanner to solve these problems.

First International Computer Programming Education Conference (ICPEC 2020).
Editors: Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões; Article No. 16; pp. 16:1–16:7

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2 New Programming Education Method

2.1 Comparison of programming education types

Programming education is divided into three areas: unplugged programming [2, 1], visual programming [3, 5], and physical programming [6, 4]. Table 1 summarizes the features of each facet of programming education when teachers conduct classes at an elementary school.

Table 1 Comparison of programming education types.

	Used items	Initial installation	Tech. knowledge required costs	Children's interest by teachers	Ease of class management
Unplugged Programming	Cards Papers	**5** Low cost	**5** Not required	**2** Lose interest easily	**5** Possible with one teacher Possible in any classroom
Visual Programming	PCs or Tablets	**3**	**3** A little required	**4** Interested	**3**
Physical Programming	PCs or Tablets Sensors, Robots	**2** High cost	**2** Strongly required	**5** Very interested	**2** Difficult to prepare

(5: Excellent; 4: Good; 3: Fair; 2: Poor; 1: NA)

It is commonly thought that programming needs to be learned on a computer, but if students are to gain a deeper understanding of the concept of the program rather than operating the program blindly, learning in the unplugged form is effective. Furthermore, unplugged programming has positive features, including low budget requirements and ease of use in the classroom. Since programming classes in lower grades are conducted in a general classroom rather than in a laboratory, there is almost no space for equipment, and it is difficult to perform visual programming and physical programming. On the other hand, unplugged programming can be handled relatively easily in a small space.

However, there is a problem that children get bored faster than with physical programming methods that use robots. Children seem to be impressed and highly motivated by physically controlling robots. Therefore, in this study, we propose a new educational method that makes use of the features of both unplugged and physical programming.

2.2 New programming education method

Figure 1 is an outline of the new programming education method. A new teaching method uses stickers and a scanner with instructions for controlling a robot car (PS: programming sticker). Each child thinks of a procedure for solving problems at his/her desk and applies a sticker to control the robot according to the procedure.

Children use the new materials in the following steps.

1. The children will be given the task written on the task sheet. For example, give the children the task of controlling a robot car in a supermarket to buy rice and curry food.
2. Children choose the ingredients to buy and think of a route to buy them efficiently. The PS is a special sticker that can be stuck or peeled off any number of times, and can be programmed by the child in trial and error.
3. Next, when the programming sheet with PS stuck is read by the overhead scanner (Fujitsu ScanSnap SV600), it is automatically coded and the control instruction is transferred to the robot car via the computer.
4. The children can check the operation of the program by running the robot car (LEGO EV3) containing the program created by themselves on the actual course.

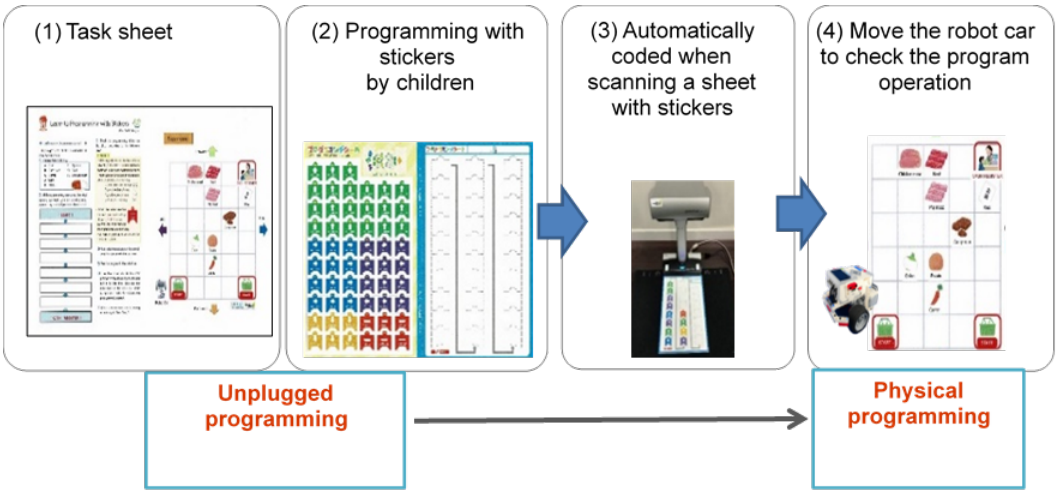


Figure 1 Outline a new programming education method using programming with stickers (PSs) and a scanner.

The operation is simple from scanning to moving the robot car, and it can be performed only by children without the help of teachers. Therefore, an elementary school teacher can give lessons in the form of 1 (teacher) vs. N (the number of children). Furthermore, unplugged programming is performed in the program creation process, and the operation check of the created program is physical programming. This has the advantage of reducing the number of devices required for the class, such as robots and personal computers.

2.3 Educational system configuration

Figure 2 shows the configuration of the educational system. A non-contact scanner (Fujitsu ScanSnap SV600[1]) was used to scan an image of the programming sticker. This scanner is suitable for scanning uneven sheets, such as programming stickers, as it does not touch the stickers during overhead scanning. A laptop computer captures the image from the scanner, identifies the JPEG image of stickers, and converts it into robot control information (JavaScript Object Notation data: JSON data). Since this image recognition is performed by the color of the sticker, even if the sticker put by the children is inclined, it can be recognized reliably. The LEGO Mindstorms EV3 was used for the robot car. LeJOS firmware[2] was installed on EV3 to realize JAVA programming with LEGO. The JSON data were sent from the computer to EV3 via a USB cable. There was no need for any expert knowledge, as all steps just require the pressing of a button.

2.4 Programming Sticker and Programming Sheet

Figure 3 shows programming stickers for lower grade elementary school children. The left side is programming stickers and the right side is a sheet to put stickers on. The PSs and the sheet are all made of paper, and the sheet surface is treated to make it easy to remove the sticker.

[1] https://www.fujitsu.com/global/products/computing/peripheral/scanners/scansnap/sv600/
[2] https://sourceforge.net/projects/lejos/

The stickers can be put on and peel off, so children can programming with stickers by trial and error. Since the children are new to programming, only five stickers were used: Straight, Right turn, Left turn, Reverse and Stop.

Figure 2 Configuration of the educational system.

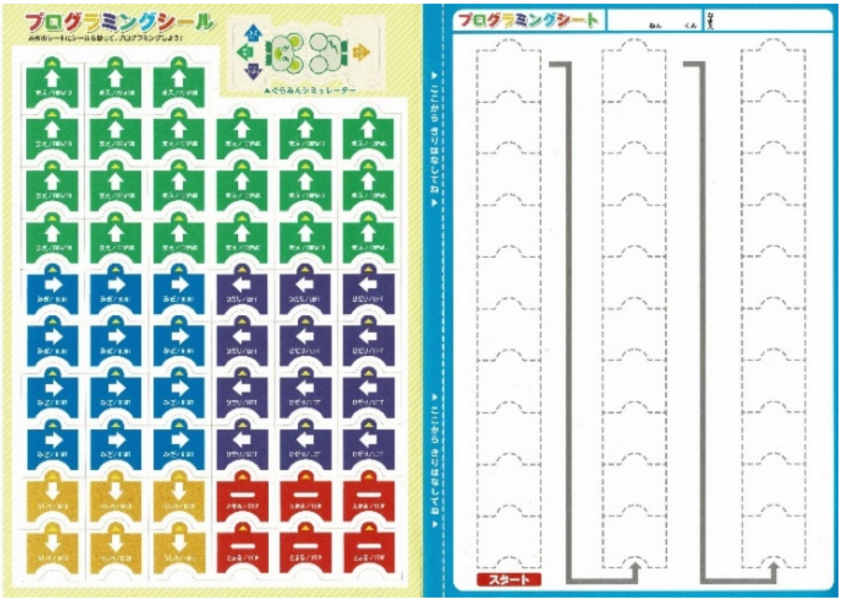


Figure 3 Programming stickers (left side) and a programming sheet (right side).

3 Trial Experiment of New Programming Education

3.1 Trial experiment in elementary school classes

Trial experiments were conducted in an elementary school using new teaching methods. The target children were 66 third-grade of Meiko Elementary School, Hakusan City, divided into two classes.

Figure 4 shows the task sheet prepared for this class. The actual task sheet used was written in Japanese. The children were tasked with buying food for rice and curry by controlling a robot in a supermarket.

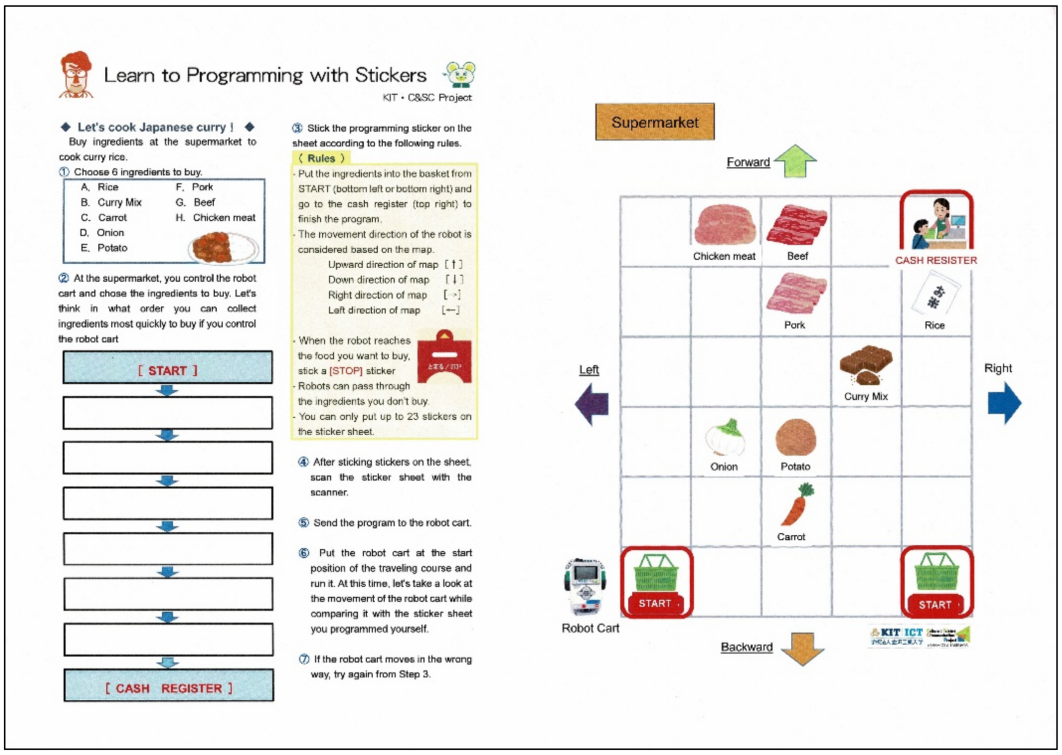


Figure 4 Task sheet, "Buy Japanese curry ingredients in a supermarket".

The task sheet and programming sticker were distributed to each student. Two sets of scanners and laptop computers, eight robots, and eight traveling courses of robots were prepared.

Figure 5a shows a picture of a child programming using PSs. Many children were able to stick PSs on the sheet freely without any assistance from teachers. Figure 6 shows a scan of the programming sheet and data transfer to the robot car. It takes only about 15 seconds from the scanning of the programming sheet to the completion of the data transfer, greatly reducing equipment usage time. Therefore, eight robot cars were enough to deal with 33 children. Figure 5b shows the robot car moving on the traveling course. By comparing the movement of the robot car with the programming sticker, the child can confirm the operation of the program he or she thought. If the child notices a mistake in the movement of the robot, the child will notice it and can re-stick the sticker.

3.2 Questionnaire after class

A questionnaire survey was conducted to confirm that the proposed teaching materials could be used. The children were 35 boys and 31 girls, and 86% of them experienced programming classes for the first time. Figure 7 shows the results for the following questions.

Q1 Was the content of this class difficult for you?
Q2 Is the programming sticker easy to use?
Q3 Is the scanner easy to use?

(a) Programming using PSs.

(b) Program operation check.

Figure 5 Images from students.

Figure 6 Scanning PSs stuck on the sheet and transferring data to the robot car.

Q4 Is the robot car easy to use?
Q5 Did you enjoy this class?
Q6 Were you interested in programming after this class?
Q7 Do you want to take programming classes again?

From the results of Q2, Q3, and Q4 questionnaires, it became clear that the teaching material system components can be used by children without problems. Additionally, more than 90% answered that they enjoyed this class, and more than 85% answered that they were interested in programming. The survey results suggest that the new teaching methods we have developed can be used for programming education for elementary school children.

4 Conclusions

A new programming educational method using stickers and scanner was described. This method combines the features of unplugged programming and physical programming, and the new teaching materials developed have excellent features compared to commercially available teaching materials, such as lower cost, use in lower grades of elementary school, and no need for ICT skills for teachers. Trial experiments were conducted on 66 third-grade elementary school students to confirm the effectiveness of the materials. It became clear that the new educational method we developed could be used for programming education for elementary school children, as evidenced by the results of the questionnaire.

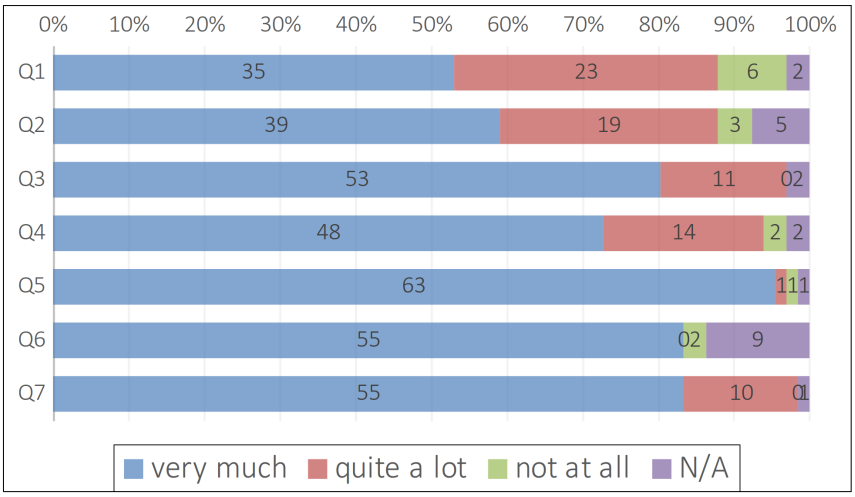


Figure 7 Results of the survey.

References

1 Christian P. Brackmann, Marcos Román-González, Gregorio Robles, Jesús Moreno-León, Ana Casali, and Dante Barone. Development of computational thinking skills through unplugged activities in primary school. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, WiPSCE '17, page 65–72, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/3137065.3137069.

2 Yvon Feaster, Luke Segars, Sally K. Wahba, and Jason O. Hallstrom. Teaching cs unplugged in the high school (with limited success). In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, ITiCSE '11, page 248–252, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/1999747.1999817.

3 Takayuki Dan Kimura, Juli W Choi, and Jane M Mack. Show and tell: A visual programming language. *Visual Programming Environments: Paradigms and Systems*, pages 397–404, 1990.

4 Pamela B. Lawhead, Michaele E. Duncan, Constance G. Bland, Michael Goldweber, Madeleine Schep, David J. Barnes, and Ralph G. Hollingsworth. A road map for teaching introductory programming using lego© mindstorms robots. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '02, page 191–201, New York, NY, USA, 2002. Association for Computing Machinery. doi:10.1145/960568.783002.

5 Orni Meerbaum-Salant, Michal Armoni, and Mordechai (Moti) Ben-Ari. Learning computer science concepts with scratch. In *Proceedings of the Sixth International Workshop on Computing Education Research*, ICER '10, page 69–76, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1839594.1839607.

6 Achille Messac. Physical programming - Effective optimization for computational design. *AIAA Journal*, 34(1):149–158, January 1996. doi:10.2514/3.13035.

7 Jeannette M. Wing. Computational thinking. *Communications of ACM*, 49(3):33–35, March 2006. doi:10.1145/1118178.1118215.

8 Jeannette M. Wing. Computational thinking and thinking about computing. In *2008 IEEE International Symposium on Parallel and Distributed Processing*, pages 1–1, 2008.

Detailing an e-Learning Course on Software Engineering and Architecture Using BPMN

Ceres Morais
INESC TEC, Porto, Portugal
Universidade do Estado do Rio Grande do Norte, Mossoró, Brasil
http://www.inesctec.pt
ceresmorais@uern.br

Daniela Pedrosa
CIDTFF, Aveiro, Portugal
University of Aveiro, Portugal
https://www.ua.pt/cidtff/
dpedrosa@ua.pt

Mario Madureira Fontes
Universidade Aberta, Coimbra, Portugal
Pontifícia Universidade Católica de São Paulo, Brasil
https://www.pucsp.br/
mario@mario.pro.br

José Cravino
CIDTFF, Aveiro, Portugal
University of Trás-os-Montes e Alto Douro, Vila Real, Portugal
http://www.utad.pt
jcravino@utad.pt

Leonel Morgado[1]
INESC TEC, Porto, Portugal
Universidade Aberta, Coimbra, Portugal
leonel.morgado@uab.pt

Abstract

We have employed BPMN diagrams to expose the foreseen teaching and learning activities of participants in an e-learning course under planning. This provided clarification of the teaching and learning actions, revealing to the educational planning team aspects which were not explicit in the lecturer's plan, such as: the level of effort for the teacher as well as for the student; specific moments when there is a need to provide feedback and motivation. We believe that this exercise constitutes a rich and helpful contribution in planning and visualization efficient for other teaching teams of computer programming courses.

2012 ACM Subject Classification Social and professional topics → Software engineering education

Keywords and phrases educational planning, BPMN, e-learning, programming courses, MVC, software engineering education

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.17

Funding This work is co-financed by national funds through FCT – Fundação para a Ciência e a Tecnologia, I.P., as part of project UID/CED/00194/2019, SCReLProg. And also by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalization - COMPETE 2020 and Lisboa 2020 under the PORTUGAL 2020 Partnership Agreement, and through the Portuguese National Innovation Agency (ANI) as a part of project CHIC POCI-01-0247-FEDER-024498.

[1] Corresponding author

First International Computer Programming Education Conference (ICPEC 2020).
Editors: Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões; Article No. 17; pp. 17:1–17:8
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Planning the educational process is key for effective course preparation [9]. However, most educational planning focuses syllabus content of the syllabus, not rendering explicit actual activities expected of teachers and students throughout the semester, even though that's what really matters and leads to learning [5]. We sought to render them explicit, a necessity in online contexts, due to their different interaction dynamics, where there is a significant challenge to achieve adequate pedagogical design that is effective [14]. For this, we revised a course plan laying out expected actions from all involved parties (teacher, students, information systems). The outcome revealed hitherto unexpected complexity in actual tasks, enabling us to decide on improvement measures. In this paper, we exemplify this approach, towards increased perception by educational planners (e.g., learning designers, teachers, program supervisors, e-learning platform managers) of the benefits of BPMN (Business Process Model and Notation) [7] to reveal which activities are expected of involved parties. This visual notation renders explicit interactions, communications, and actions of the parties, enabling reflection upon it to identify improvement targets: bottlenecks, workloads, decision-making with insufficient information, unforeseen tasks, etc. The course under scrutiny was "Software Development Laboratory", part of the undergraduate program on Informatics Engineering at Universidade Aberta, Portugal's public online learning university. Within the next two sections, we provide an overview BPMN concepts and previous efforts to employ it in education and the course context. In subsequent sections, we provide two exemplary cases of educational activities described with BPMN, through which reflection was enabled upon aspects unforeseen in traditional course planning. We conclude by highlighting how this process can also contribute to ascertain informational and technical support needs of the involved parties. The identification of such needs enables individuals to acts to better manage the associated workload and may contribute to the development of more effective learning and teaching support tools.

2 Related work: BPMN in educational planning

In online education, students are typically working at their own pace through the materials, and interactions are generally asynchronous. What makes it challenging is that the teacher must depart from usual experiences of physical, face-to-face interactions towards new teaching methods, which require greater focus on time management. This implies planning learning activities that rely on students' autonomy and initiative - and that align with an older target audience (typically mid-20s onwards, over several decades of age range), and that are deployable with large virtual class sizes [8]. These stricter educational planning needs can benefit from BPMN, a process-description graphical notation [7]. It enables exposing roles and interactions of stakeholders with learning activities [2], supporting answering questions such as "Why is it done? By whom? Where? When? How is it performed?". Its use in modelling and managing e-learning processes aims to facilitate their development and maintenance [12]. The literature provides a diversity of examples: planning software engineering hands-on classes for maturity and appraisal process analysis [13]; analysing interactions and activities in learning processes [1]; or visualizing train-the-trainer sessions in blended environments [6]. In general, the ability of BPMN to support a clear understanding of the interconnection of learning stakeholders throughout processes enables the identification of improvement potential, such as discussing possible intervention points and reasonable activities [3].

3 The course context

The course "Software Development Laboratory" (LDS, Portuguese-language acronym) has an online asynchronous e-learning format, in the Moodle platform, during the 2nd semester of the 2nd year of the Informatics Engineering undergraduate program of Universidade Aberta (UAb), Portugal, over 12 academic weeks. Its goal is to scaffold undergraduates in their transition from novice programmers into proficient programmer, pursued over a six-topic syllabus. Since it is asynchronous, no specific schedule exists for fulfilling activities. Students complete them at their own pace and within their own schedule, within deadlines. Discussion with the teaching staff and colleagues is also done asynchronously. This temporal flexibility is mandated by UAb's pedagogic model, which states it as a cornerstone [11]. The rationale includes the fact that UAb's students have typical demographics of online education: 30-50 year-old adults, with scheduling constraints of active professional careers and the need to care for children/teenagers or older relatives; and disseminated across the globe, with wildly varying timezones. Thus, during the two-week span of each topic, the teacher must track and encourage progress and interaction, since any procrastination until the final days of the span will lead to lack of opportunities for teacher and/or peer feedback. LDS has been the field for pedagogic experimentation of the software engineering didactic approach SimProgramming [10], created for a physical classroom course with similar goals.

4 Exemplary cases in BPMN

BPMN modelling revealed unexpected complexities in apparently simple tasks in the LDS course. Here we present two exemplary cases, resulting from interactive collaborative design of the BPMN diagram, involving the course lecturer, a didactics researcher, and two educational technology researchers.

4.1 Case one: viewing a slideshow

Viewing a slideshow is possibly as plain a situation as any one could expect in e-learning: the course provides it, students watch it, the teacher checks progress, and the learning situation is complete. However, process analysis reveals a more complex - and time-demanding - reality. Fig. 1 presents this activity in BPMN, focusing on the teacher's activities (the student tasks likely trigger further actions, such as studying references materials, etc.). There are in all 16 activities, distributed between the teacher, the learning management platform, and the students. The plainer view is reflected by the leftmost activities: the students watch the slideshow, the learning platform collects data on this, generates graphics and reports, and the teacher views them. If a student views the slideshow timely, its simple task is to report completion and later check if due credit/assessment was granted. The diagram however reveals the complexity and workload hidden in the deceptively simple earlier sentence "the teacher checks progress" - which comprises 10 of the total 16 activities. Firstly, students must be encouraged to view the slideshow, in case they haven't already. This is a Goldilocks scenario: not too early, not too late, not too often, just the right amount [4]. This takes place over two weeks, a period during which two reminders or encouragement should occur: first after five days (before the first weekend, prime study period for older adults). The second two days before the end of the period, as a final, personalized effort to encourage participation From the teacher's perspective, these two specific moments involve a specific workload. Actual reminders must be set. Then on the first moment the class encouragement must be posted, and on the second moment the teacher needs to check which students haven't

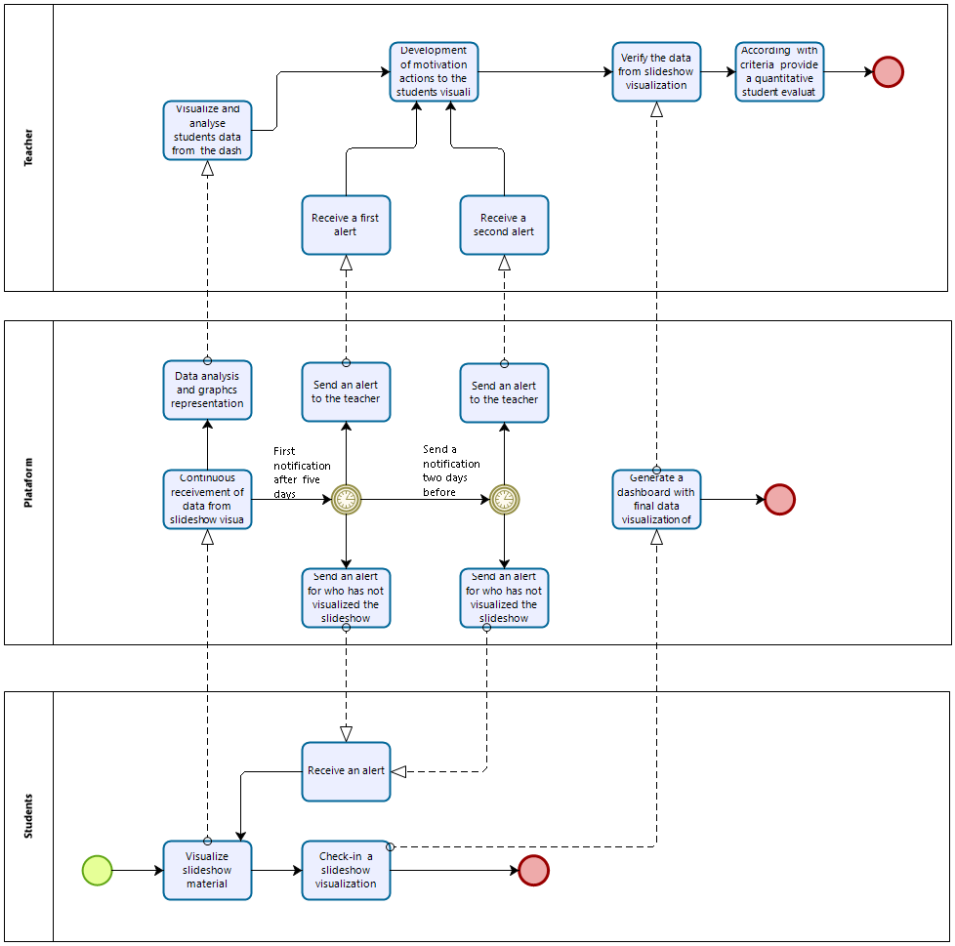


Figure 1 Activity diagram of slideshow visualization process.

yet viewed the slideshow, to send personalized messages. This workload emerges from the BPMN diagram as moments to reserve appropriate time and avoid forgetting. Further, the diagram includes a larger task amidst those moments: "Development of motivation actions to the students visualization". Those motivational actions are diverse, and may include monitoring viewing patterns, comparing them with individual students' class participation habits, considering other parallel tasks in the same course for the same period, etc. The BPMN has framed the scope of this task as that occurring between set reminders, clarifying for the teacher reflection and planning needs that it involves.

4.2 Case two: discussing the syllabus

The PUC document (course plan, Portuguese-language acronym) is similar to a syllabus, providing goals, content, deadlines, assessment methods, and study workload plan for the semester. All students are required to debate it asynchronously for clarifications or changes. Managing such interactions is a demanding task for the teacher, having to track all student postings or lack thereof, establish whether feedback and encouragement are required or not. It is not a clear-cut decision: students may be straying too far off from effective approaches, or even pursuing a misconception, which may require prompt intervention; or they may need

to develop autonomy and feel encouraged by peers. Students may not be posting due to a variety of reasons, requiring different kinds of encouragement. They could be planning on working over the weekend, in which case a simple Friday reminder would be adequate; or they could be overwhelmed by preliminary readings, in which case a more personal tutoring style would be adequate; or yet other situations. All these need a teacher's time and attention, which when managing several discussions and courses, can be overwhelming. An example of BPMN analysis of discussion activities is shown in Fig. 2 for the PUC document. Since all students should discuss, participation tracking is geared towards issuing alerts and providing context-relevant data and activity support to different stakeholders. Most students will not participate right away. While some may have a moonlighting schedule, others concentrate coursework over weekends, with evenings devoted only to the most pressing activities. However, if no students participate at all, this will be discouraging: it is necessary to "break the ice" of the dynamics. So for the first few days, the teacher focuses on the overall participation level: if it remains low, issues a global encouragement. Nearing the weekend prime study time, if global participation remains low, a reminder is appropriate. By Monday (day 8), the teacher checks for participants that have not yet acted and sends them individual encouragement messages. Also, per UAb's pedagogic quality standards, any feedback must be provided within two working days, so by day 10 the teacher must provide any due feedback. As the final weekend nears, the final opportunity for timely participation in the activity, the teacher should escalate and send personalized messages to inactive students, considering each individual's status and history, with encouragement, advice on possible consequences of not reading the course plan (lack of awareness of workload, losing an opportunity to contribute to assessment methods, etc.). Throughout, the teacher analyses participation data, to decide on the course of action. We identified three continual data analysis tasks: overall participation; individual participation; queries feedback needs. These are encapsulated in Fig. 2 as "Data analysis". A fourth data analysis task occurs on day 12: in order to provide personalized feedback, the teacher must analyse each non-participating individual's status and history.

5 Discussion

The starting point for the modelling of the educational process within the LDS course were the tasks foreseen for the teacher and students, in the original (traditional) planning. During the process of BPMN modelling of the course, gaps organically emerged: i.e., situations where decisions had to be made but no explicit teacher action was foreseen beforehand, and similar situations. Either the visual diagramming process would not be able to proceed or it would display empty sections without actions between decision-based events. That is to say, the analysis required for BPMN identified shortcomings in the earlier outcome of the traditional course planning. This clearer perspective on the teacher's actions that achieved through the process of BPMN diagramming enabled us to check with more rigour the teaching workload required to maintain the intended pedagogical intervention quality of the planned course activities. This outcome is consistent with that of the Brazilian team reported in the background section [12]. Also, this clearer perspective on the teaching workload can empower teachers to adjust their efforts or develop customized tools to support that workload. Specifically, since decision-making moments are explicit, this contributes to identifying which data must be collected in order to make those decisions. Also, since outcomes are also explicit, it is possible to prepare in advance some supporting instruments, such as feedback templates, rubrics, etc. From the analysed cases, for instance, we can identify the following data needs and design supporting instruments:

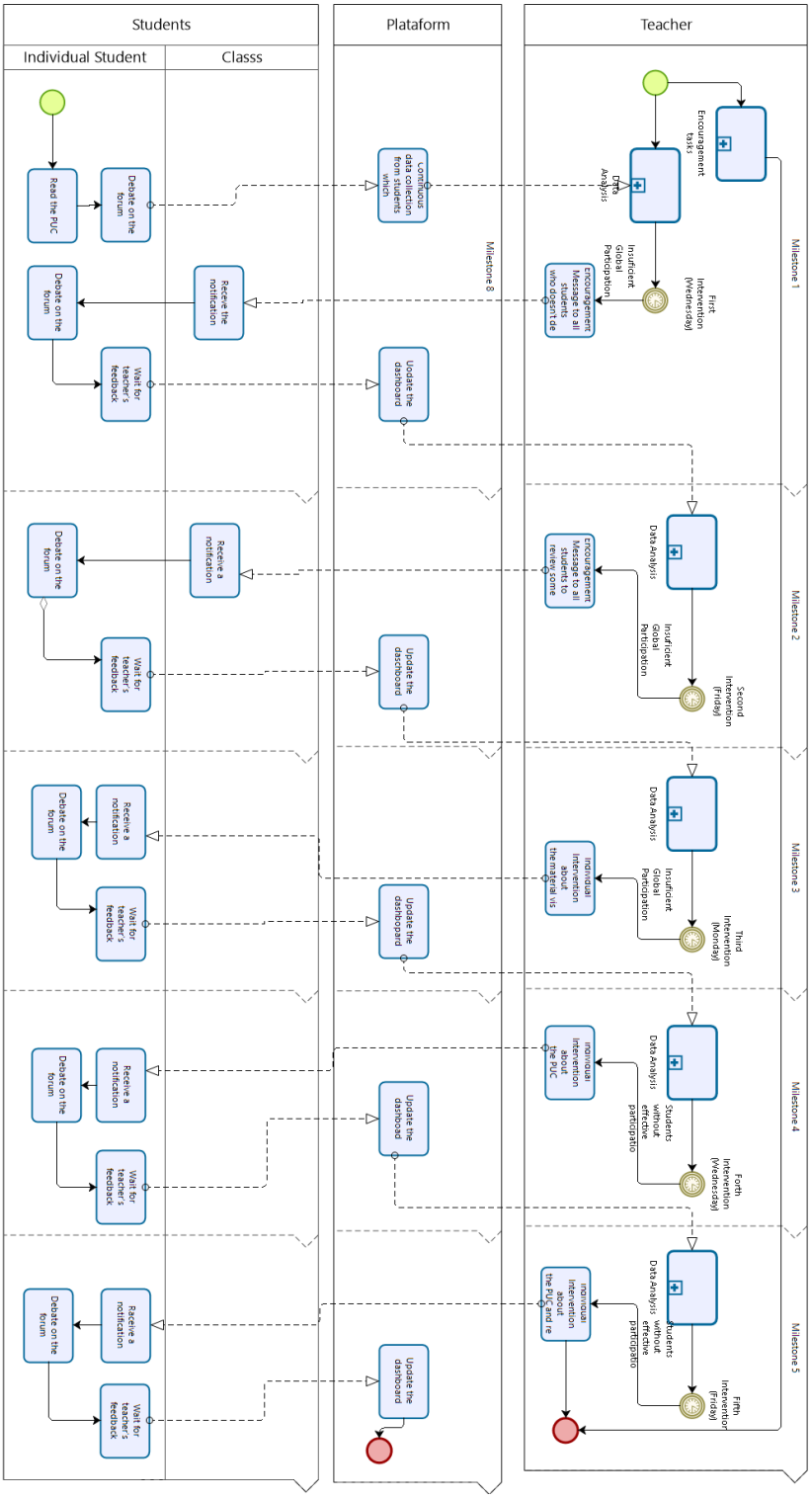


Figure 2 BPMN diagram of tasks for visualization and participation in PUC forum.

- data need: overall participation level;
- supporting instrument: reminders upon deadline of scheduled tasks;
- supporting instrument: templates/rubrics to expedite feedback preparation and drafting;
- data need: list of students requiring day-8 individual messages;
- data need: list of students requiring day-12 personalized messages, containing each individual's status and history.

These data needs and supporting instruments can be developed independently by the teacher, resorting to various freely available tools. They can also be employed by software development teams to craft automated support features in the learning platforms. The crucial aspect was their identification, achieved via the modelling process. Other data needs, expectations, assumptions, and support needs may emerge through model development, since it enables the identification of collaboration possibilities and requirements between the involved parties, their connections, the activities in which they participate, and their responsibilities. As an example, when analysing data provided by the platform, the teacher may establish the need to respond to students queries, check the overall level of student participation within an activity, or identify individual participation. With those data, the teacher will be able to decide, throughout the course, on whether encouragement is necessary, whether teamwork is actually progressing, if there is mutual cooperation, and other factors relevant for active strategies such as SimProgramming [10]. Finally, BPMN modelling of course modules may contribute to streamline modelling of other modules or courses, since patterns may emerge to be replicated readily. This streamlining may contribute towards quick dissemination of these benefits to entire courses, programmes or even institutions.

6 Conclusions

The use of BPMN revealed the activities of the various parties involved in the course: teacher, students, and technological platform. Revealed aspects, tasks, and workload, often implied, not explicit. This makes it a promising tool and approach to support the planning process. The BPMN specification of the course planning helped identify and define concrete occasions for critical interventions: when to provide specific, individual and encouragement feedback, and which data to support that is available for the teacher and from the platform. This enhanced perspective exposed an unexpected level of complexity amidst the interactions between the various parties and their level of coupling. The clarification of the processes that occur enabled us to design strategies to promote self-regulation and co-regulation, identifying required interventions, their opportunity, and the involved parties.

7 Future work

Reflecting upon this experience using BPMN, we are considering its potential as a tool for identifying specific alternatives for better teaching action. It may provide a significant contribution towards didactic and operational re-engineering of courses (i.e., leading to the development of different pedagogical approaches, tools, and interventions), and towards more adequate, more effective educational planning. We are also considering its potential to support the implementation of self-regulation and co-regulation strategies, by supporting students' academic participation and study planning, which contribute towards success. Thus, we suggest expanding this work by using BPMN to specify higher-order educational tasks, more complex and demanding at the cognitive and operational levels than the ones presented herein. Such research may be fruitful for identifying processes and interventions that hitherto where not clearly foreseen in traditional planning.

References

1. S. Azouzi, Z. Brahmi, and S. Ghannouchi. Customization of multi-tenant learning process as a service with Business Process Feature Model. *Procedia Computer Science*, 126:606–615, 2018. doi:10.1016/j.procs.2018.07.295.
2. S. Azouzi, S. Ghannouchi, and Z. Brahmi. Modeling of a Collaborative Learning Process with Business Process Model Notation. In *Digital Economy. Emerging Technologies and Business Innovation*, volume 290, pages 95–104. Springer International Publishing, Cham, 2017. doi:10.1007/978-3-319-62737-3_8.
3. P. Bitzer, M. Söllner, and J. Leimeister. Design Principles for High-Performance Blended Learning Services Delivery: The Case of Software Trainings in Germany. *Business & Information Systems Engineering*, 58(2):135–149, April 2016. doi:10.1007/s12599-015-0403-3.
4. S. Brookhart. *How to give effective feedback to your students.* Association for Supervision and Curriculum Development, Alexandria, Va, 2008.
5. J. Cravino. Planning teacher mediation in science and technology lessons. In J. B. Lopes, J. Cravino, E. de Souza Cruz, and A. Barbot, editors, *Teaching science: contributions of research for planning, practice and professional development*, University teaching and faculty development, pages 45–59. Nova Science Publishers, New York, 2017.
6. N. Faizan, M. Gottlieb, A. Loffler, M. Utesch, and H. Krcmar. State-of-the-Art to Measure the TPACK Level of Trainees in Higher Education to Increase the Learnability of the Train-The-Trainer (TTT) Sessions. In *2019 IEEE Global Engineering Education Conference (EDUCON)*, pages 384–391, Dubai, United Arab Emirates, April 2019. IEEE. doi:10.1109/EDUCON.2019.8725056.
7. OMG Object Management Group. Business Process Model and Notation (BPMN), Version 2.0. Technical Report formal/2011-01-03, OMG - Object Management Group, January 2011. URL: http://www.omg.org/spec/BPMN/2.0.
8. M. Kebritchi, A. Lipschuetz, and L. Santiague. Issues and Challenges for Teaching Successful Online Courses in Higher Education: A Literature Review. *Journal of Educational Technology Systems*, 46(1):4–29, September 2017. doi:10.1177/0047239516661713.
9. J. Kettunen. Strategy and Quality Maps in Higher Education. *US-China Education Review*, 8(2):149–159, 2011.
10. D. Pedrosa, J. Cravino, L. Morgado, C. Barreira, R. Nunes, P. Martins, and H. Paredes. Simprogramming : the development of an integrated teaching approach for computer programming in higher education. In *INTED2016 Proceedings - 10th International Technology, Education and Development Conference March 7th-9th, 2016 — Valencia, Spain*, pages 7162–7172, Valencia, Spain, March 2016. IATED Academy. doi:10.21125/inted.2016.0699.
11. A. Pereira, A. Quintas-Mendes, L. Morgado, L. Amante, and J. Bidarra. *Universidade Aberta's pedagogical model for distance education: a university for the future.* Universidade Aberta, Lisbon, Portugal, 2008. URL: http://hdl.handle.net/10400.2/2388.
12. M. Pereira. Model for production of teaching material: the use of bpmn notation in distance learning courses. *Review of Administration and Innovation - RAI*, 8(4):45–66, January 2012. doi:10.5773/rai.v8i4.898.
13. F. Siqueira, G. Barbarán, and J. Becerra. A Software Factory for Education in Software Engineering. In *2008 21st Conference on Software Engineering Education and Training*, pages 215–222, Charleston, SC, USA, April 2008. IEEE. doi:10.1109/CSEET.2008.10.
14. B. Trammell and C. LaForge. Common Challenges for Instructors in Large Online Courses: Strategies to Mitigate Student and Instructor Frustration. *Journal of Educators Online*, 14(1):10, 2017.

Game-Based Coding Challenges to Foster Programming Practice

José Carlos Paiva
CRACS – INESC-Porto LA, Portugal
DCC – FCUP, Porto, Portugal
jose.c.paiva@inesctec.pt

José Paulo Leal
CRACS – INESC-Porto LA, Portugal
DCC – FCUP, Porto, Portugal
https://www.dcc.fc.up.pt/~zp
zp@dcc.fc.up.pt

Ricardo Queirós
CRACS – INESC-Porto LA, Portugal
uniMAD – ESMAD, Polytechnic of Porto, Portugal
http://www.ricardoqueiros.com
ricardoqueiros@esmad.ipp.pt

Abstract

The practice is the crux of learning to program. Automated assessment plays a key role in enabling timely feedback without access to teachers but alone is insufficient to engage students and maximize the outcome of their practice. Graphical feedback and game-thinking promote positive effects on students' motivation as shown by some serious programming games, but those games are complex to create and adapt. This paper presents Asura, an environment for assessment of game-based coding challenges, built on a specialized framework, in which students are invited to develop a software agent (SA) to play it. During the coding phase, students can take advantage of the graphical feedback to complete the proposed task. Some challenges also encourage students to think of a SA that plays in a setting with interaction among SAs. In such a case, the environment supports the creation and visualization of tournaments among submitted agents. Furthermore, the validation of this environment from the learners' perspective is also described.

2012 ACM Subject Classification Applied computing → Interactive learning environments; Applied computing → E-learning

Keywords and phrases games, automatic assessment, graphical feedback, programming, learning, challenges

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.18

Funding This work is financed by National Funds through the Portuguese funding agency, FCT – Fundação para a Ciência e a Tecnologia, within project UIDB/50014/2020.

1 Introduction

The amount of time dedicated to practice in programming classes is scarce considering the quantity of knowledge that students need to assimilate. For instance, in a regular sixteen-week semester, introductory programming courses often do not have more than twelve programming assignments [2], which is both too much for teachers to provide meaningful individualized feedback to students and too little for novices to master programming skills. Consequently, it is necessary to provide students with the right tools and resources to increase practice time and maximize its outcome.

Automated assessment tools were found useful to grant instant, teacher-free, and effortless feedback for students on attempts to solve exercises [1]. These tools can provide both static and dynamic program analysis. Static analysis involves the compilation of the source code whereas dynamic analysis evaluates the compiled source code, typically providing some input and comparing its output to the solutions' output. The importance and benefits of such approaches in learning are well-documented in the literature [6, 5]. Yet, since these tools are primarily developed for programming contests, their feedback is usually insufficient for learning as is their means to engage and retain learners.

Notwithstanding, tuning dynamic analysis of programs can play a key role in motivating practitioners. For instance, CodeRally [8] and RoboCode [7] are two popular games among programmers in which players are Software Agents (SAs), developed by them, that compete against each other in a 2D environment. The SAs' owners then visualize the graphical feedback of the game unfolding. These games are used both by novice programmers to learn Object-Oriented Programming concepts and more advanced programmers for improving skills while taking pleasure out of it. Some teachers introduced them in programming classes with quite success [3], but only in certain topics as the complexity and costs of creating this kind of games undermines their adoption for teaching other concepts than those they were designed for.

This paper presents Asura, an automated assessment environment for game-based coding challenges, that aims to engage students while working on their programming assignments to increase the time spent on and the outcome of their programming activities. To this end, it fosters students' motivation by challenging them to code SAs. The feedback provided by the evaluation environment consists of a movie displaying the behavior of the SA during the game. These challenges are boosted with competition, among all submitted SAs, in the form of tournaments such as those found on traditional games and sports, including knockout and group formats. The outcome of the tournament is presented on an interactive viewer, which allows learners to navigate through each match and the rankings. On the teachers' perspective, Asura provides a framework and a command-line interface (CLI) tool to support the development of challenges. The validation of this authoring framework has been already conducted and described in a previous empirical study [11]. Hence, this paper focuses on the learners' perspective of Asura.

The remainder of this paper is organized as follows. Section 2 reviews the state of the art on platforms and tools that make use of challenges similar to those of Asura to engage programming learners. Section 3 provides an in-depth overview of Asura, including details on its architecture, design, and implementation. Section 4 describes its validation regarding the growth in motivation and, consequently, practice time. Finally, Section 5 summarizes the main contributions of this research.

2 State of the Art

The idea of fostering programming students' motivation through games is not new, neither that of challenging novice programmers to develop an SA to make decisions as a player of a game. The novelty of Asura is that it aims to make the creation of game-based coding challenges, where the learner has to code an SA for a game, simple enough to allow their use in educational contexts for teaching specific programming concepts and increase practice. Therefore, this section introduces some platforms and tools that also provide game-based programming challenges with graphical game-like feedback.

2.1 CodinGame

CodinGame[1] is a web-based platform that proposes several puzzles for learners to practice their coding skills. Most of these puzzles require the user to develop an SA to control the behavior of a character in a game environment, and offer game-like graphical feedback. The SA programmed by the player must pass all test cases (public and hidden) to solve the puzzle. Players can choose one of the more than twenty programming languages available to write their SA, or even solve it in different languages. Once the exercise is solved, players can access, rate, and vote on the best solutions.

Solving these game puzzles and awarding votes on solutions contributes to the leaderboards, level, and badges of the user. There are also contests from time to time in which the player can receive real-world rewards, by defeating other players' agents in a match or being the first to solve all problems. Another mode that CodinGame provides is the *Clash of Code*, where a player competes against other players to be the first to submit the best solution to an exercise. These exercises typically last for five minutes and can be authored by the community, although only text-based test cases are supported in this case.

Even though it has a large variety of challenges, this platform is not adequate to a classroom setting as educational resources are scattered, lacking a proper order for learning. Furthermore, it is a commercial platform and does not allow external persons to author challenges or modules.

2.2 SoGaCo

SoGaCo (Social Gaming and Coding) [4] is a scalable cloud-based web environment that evaluates competitive SAs, developed either in Java or Python, for 2-player board games. The user interface of SoGaCo has two distinct views: one for editing code and another to test the agents. The latter contains three panels on the left with the user bots, built-in bots, and shared bots from other users, one panel on the center which displays the graphical feedback, and three panels on the right to control the step-by-step visualization of the graphical feedback, show the score, and write game captions.

The user starts coding the SA from a skeleton provided by the game author. During the development, he/she can test the bot against any bot present in one of the three bots' panels. After completing it, the single bot address (URL) can be shared with other peers to either compete against it or see its code. The modular architecture of SoGaCo supports different games, but there is no known framework or standard form to develop games for SoGaCo. Nevertheless, it already contains four board games, namely PrimeGame, Mancala, Othello, and 5-in-a-row.

This environment, to the best of the authors' knowledge, is the most similar with Asura. However, it only supports 2-player board games and does not provide any framework to develop new challenges or running tournaments.

3 Asura

Asura is an automatic assessment environment for game-based programming challenges. Its main goal is to provide a means to increase time dedicated to programming practice by engaging students with intrinsic motivators of games while requiring teachers a comparable effort to that of creating a traditional programming problem. Hence, this environment

[1] `http://codingame.com`

enables students to enjoy learning activities using unique features of games, such as graphical feedback, game-thinking, and competition, while including a set of tools designed to support authors during the process of creating Asura challenges, such as a framework and a CLI tool.

This environment follows a multi-component architecture, as depicted in the UML diagram of components of Figure 1, composed of three individual components, named Asura Viewer, Asura Builder, and Asura Tournament Manager, and a component designed as a plugin for an evaluation engine, Asura Evaluator. In this architecture, Mooshak [9] has been selected as the evaluation engine that will connect with Asura Evaluator. Mooshak is a web-based automatic judge system that supports assessment in computer science. It can evaluate programs written in several programming languages, such as Java, C, C++, C#, and Prolog. Furthermore, its current version includes a pedagogical environment, named Enki [12], which blends assessment (i.e., exercises) and learning (i.e., multimedia and textual resources) in a user interface designed to mimic the distinctive appearance of an IDE. This makes it a perfect fit to also integrate both the Asura Viewer on Enki and the Asura Tournament Manager on the administration interface. Furthermore, the Java ARchive (JAR) of the challenge, produced on the Asura Builder, can also be easily imported on the administration interface and used in dynamic analysis.

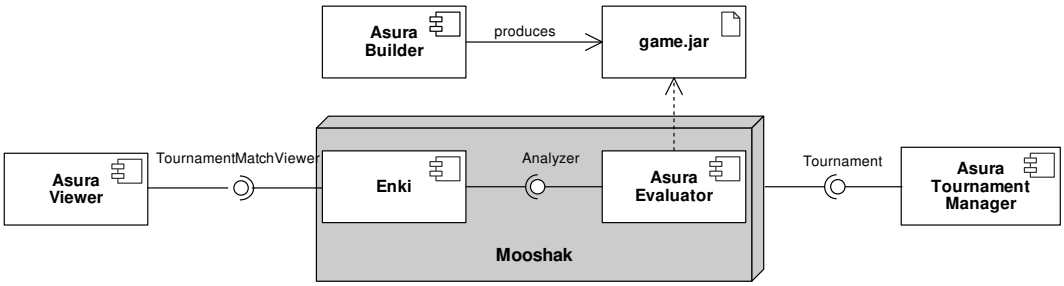


Figure 1 Architecture of the proposed ecosystem of Asura.

3.1 Asura Builder

The Asura Builder is a standalone component composed of multiple tools dedicated to the authoring of game-based coding challenges. It includes a Java framework that provides a game movie builder, a general game manager, several utilities to exchange complex state objects between the manager and the SAs as JSON or XML, and general wrappers for SAs in several programming languages. The framework is accompanied by a CLI tool to easily generate Asura challenges and install specific features, such as support for a particular programming language or a standard turn-based game manager. Even though the authors are required to program the challenges in Java, players can use their preferred programming language to code their SAs.

Since graphics concentrate much of the effort in game development, Asura Builder introduces the concept of game movie, defines a JSON schema to describe it, and provides a builder for objects adhering to it. A game movie consists of a set of frames, each of them containing a set of sprites together with information about their location and applied transformations, and metadata information. The movie should be produced while the game unfolds with the support of the builder, which, in addition to having methods to construct each frame with ease, includes methods to deal with SA evaluation and premature game termination.

The core of an Asura challenge is the game manager, which among many tasks should ensure that the game rules are followed, decide which player takes the next turn, and declare the winner(s). Asura Builder provides an abstract game manager containing all the generic code, such as the initial crossing of I/O streams, handlers for timeout and badly formatted input, automatic (de)serialization of JSON or XML to Java objects, and a "contract" for the object that manages the game state. The author only needs to implement the specific parts of the challenge being created, translating mutations of the game state into frames of the game movie.

Moreover, the framework also allows to define wrappers for the SAs. A wrapper consists of a set of methods that aim to give players an higher level of abstraction so that they can focus on solving the real challenge instead of spending time processing I/O or doing other unrelated tasks. They can also be used to increase or decrease the difficulty of the problem by changing the way that the SA interacts with the game, without modifying the game itself. For instance, a wrapper for a Go player might define a method `addStone(x: int, y: int)` to add a stone of the player in position (x, y) as well as a method `lastStone(): object` to get the last move of the opponent, in order to alleviate the hardness of the exercise. Since there are actions common to players of any game (e.g., communication with the manager), those were included in global wrappers. Hence, authors of challenges will only develop game-specific wrappers if they intend to provide a different abstraction layer to the learners.

3.2 Asura Evaluator

Mooshak's evaluation engine follows a black-box approach to grade a submission according to a set of rules while generating a report of the evaluation for further validation from a human judge. The assessment process is twofold, comprising static analysis, which checks for integrity of the source code of the SA and produces an executable program; and dynamic analysis, that involves the execution of the program with each test case loaded with the problem and the comparison of its output to the expected output.

Asura Evaluator inherits the static analysis from Mooshak, only attaching the global and game-specific SA wrappers present in the game JAR file to the command-line. However, the dynamic analysis is completely re-implemented. Instead of test cases based on input and output text files, Asura Evaluator receives as input a list of paths to opponents' submissions. The type of evaluation either a validation or a submission determines the source of the competitors, in case of multiplayer games. Validations do not count for evaluation purposes but are rather a means for the learner to experiment how his/her SA behaves in a match against any existing SA. The opponents are selected by the learner itself from a list containing all the last accepted SAs from the students as well as the control SAs included by the author. On the contrary, submissions are considered as attempts to solve the challenge and as such are evaluated equally for all participants. In this case, the opposing contestants are the control SAs provided by the challenge author.

The opponents' SAs are submissions already compiled and, thus, the component only initializes a process from the compiled sources. After that, it organizes matches containing the current submission and a distinct set of the selected opponents' submissions. The length of this set depends on the minimum and maximum number of players per match, which are specified by the game manager. At this point, the evaluation proceeds on an instance of the specific game manager, which is instantiated from the JAR. The game manager receives the list of player processes indexed by the player ID and crosses the input and output streams of each of these processes with itself. This allows the game manager to write state updates to the input stream of the SA and read actions from it output stream, as typical I/O operations

from the SA perspective. Hence, the execution of the game is completely controlled by the game manager, which is responsible for keeping the SA's informed about the state of the game, querying the SAs for their actions at the right time, ensuring that the game rules are not broken, managing the state of the game, and classifying and grading submissions.

At the end, the statuses obtained from the matches containing the observations, mark, classification and feedback are compiled into a single status which is added to the submission report, and sent to Enki.

3.3 Asura Tournament Manager

The Asura Tournament Manager is a Java library for organizing tournaments among SAs submitted and accepted on an Asura challenge. Tournaments are optional and aim to give a final objective to students by inviting them to engage in a contest realized at the end of the submission time. Hence, the challenge is not just about solving the problem, but also to prepare the SA to win a final competition. During the preparation phase, students can do "friendly" matches (i.e., validations) against any previously approved SA from each other to get an idea of what they can expect to achieve in the tourney. After this phase, instructors can use a wizard embedded in Mooshak's administration user interface to setup the tournament.

Tournaments follow similar models to those found on traditional games and sports competitions. They can have several stages, each of them arranged according to one of the available tournament formats: round-robin, swiss system, single elimination, or double elimination. A stage is composed of a series of rounds in which each competitor either participates in a single match or has a bye (i.e., advances directly to the next round of the tourney in the absence of assigned opponents). Finally, matches are the atom of a tournament. They are generated based on the tournament format, one after another, executed on the Asura Evaluator, and its outcome sent back to the Asura Tournament Manager. The result of a match is a list containing the points obtained by each player as well as any additional features that could be used as tiebreakers, either for the match or rankings.

The interaction with the library is done through the interface `Tournament`, which provides a sequential and seamless way to run the tournament. Firstly, the mandatory metadata of the tournament such as the title, game, and participants should be provided. Then, the stages are added, configured, and started one by one. The configuration options of a stage depend on its format and may include the number of players per match, the minimum number of players per group, the number of qualified players, the maximum number of rounds, the type of result of a match (e.g., win-draw-loss or position-based), and tiebreakers both for rankings and matches. Once the stage has started, the next match to execute can be obtained, activating the wait mode until the result of the match is submitted and processed. At the end, the output of the Asura Tournament Manager is a JSON object complying to the tournament JSON schema.

3.4 Asura Viewer

Asura Viewer is the component responsible for displaying graphical feedback to learners, both in single matches and in tournaments. It consists of a Google Web Toolkit (GWT) widget that transforms the provided JSON either into a movie of a match or an interactive view of the tournament, according to the schema it adheres.

The match mode is where learners can see how their SAs behaved during the match. It presents the JSON output produced during the evaluation of the submission or validation as a dynamic movie, only distinguishable from a game in the fact that it can be pushed back and forth. The widget, presented on the left of Figure 2, mimics that of a media player

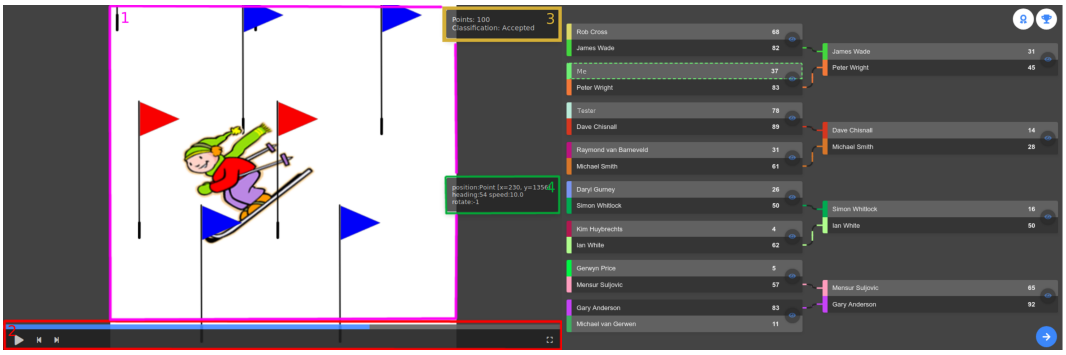


Figure 2 Asura Viewer modes. On the left, the match mode with Slalom Skier game (distinct areas highlighted in different colors). On the right, the brackets view of the tournament mode.

including a slider, a play/stop button, buttons to navigate through the current playlist, and a full-screen button in the control toolbar (red area), a box to show the current status (yellow area), a box to display debugging messages of the SA (green area), and a canvas where the movie is drawn (pink area).

The tournament mode, presented on the right of Figure 2, consists of an interactive user interface which allows students to navigate through the stages of the tournament, visualize specific matches or the whole course of a player, and check the rankings of each stage. It comprises a navigation menu on the bottom right corner to swap the current stage, a contextual menu on the top right corner to switch between standings view (either final or relative to the current stage) and brackets view, and the main area where matches and ranking appear.

4 Validation

An experiment was conducted to validate the acceptance of Asura by learners as well as its effectiveness both in motivating students to increase the time that they dedicate into programming practice and in maintaining or improving the knowledge acquisition and retention rates of traditional programming exercises. This experiment took the form of an open online learning course about the novel features introduced by version ECMAScript 6 (ES6) of JavaScript. The course has been announced to undergraduate students registered in the Web Technologies classes in the past semester, which provided them with some background on JavaScript but not on ES6 features.

A total of 10 students enrolled in the course, of which 1 was female. These students were randomly divided into two groups of 5, control (1 female) and treatment. Each of the groups made a separate branch of the course with the same expository resources but different evaluative resources. The expository resources are lecture notes about the concepts of ES6, including variable declaration, object and array destructuring, arrow functions, promises, and classes, and a compiled ES6 cheat sheet. The evaluative resources are either International Collegiate Programming Contest (ICPC)-like problems (control branch) or Asura challenges (treatment branch). At the end of the course, students from both groups were invited to do an exam composed only of ICPC-like problems to assess the knowledge acquired during the course.

The Asura challenges are different chapters of the same game which is a remake of Asteroids, an arcade space shooter released in November 1979 by Atari, named War of Asteroids. This remake keeps most of the original gameplay of Asteroids, but adds a number

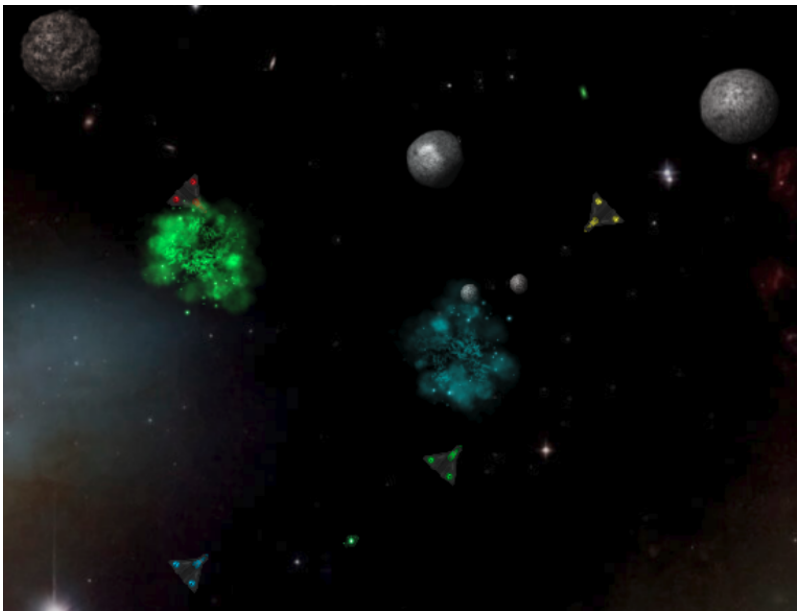


Figure 3 War of Asteroids. Screenshot of a game with four ships competing against each other.

of features; improves graphics (see Figure 3); and replaces the input controls with a program. The game is now played by up to 4 contestants, which replace the saucers. Furthermore, the ships have two additional commands that can be used: activate the energy shield and fire bombs. The ways of earning points as well as the number of points awarded per accomplishment were also modified. The game ends after 10000 units of time (frames) or when a player gets alone in the space.

The game-specific wrapper provides SAs with 7 commands, particularly, `thrust()` thrusts the ship, `steerLeft()` adds -4 degrees to the heading of the ship, `steerRight()` adds 4 degrees to the heading of the ship, `shield()` activates the shield, `firePrimary()` fires a bullet, `fireSecondary()` throws a bomb, and `log(message)` logs a message. Some of these commands are not available in the first chapters, but revealed during the course.

4.1 Results and Analysis

The data gathered during the experiment consists of usage data and questionnaire responses. The usage data is automatically captured by Mooshak 2 into the activity log based on every request sent to the server. This enables the extraction of several metrics such as the number of submissions, number of validations, date and time of activity, and submissions' results. The questionnaire is based on the Lund's model [10], including one section per metric (i.e., *Usefulness*, *Ease of Use*, *Ease of Learning*, and *Satisfaction*) with questions to classify sentences in a 7-value Likert scale and an additional section with free-text questions to collect students' feedback about Asura regarding weaknesses, strengths, and points of improvement.

The questionnaire is part of the exam to guarantee that only students who complete their course and ask for the exam would fill it in. Two of the students (one from each group) have not taken the exam and, thus, they did not answer the survey. The remaining students have finished both the exam and the questionnaire. The outcome from the questionnaire is presented in Figure 4 separated by group, control on the left and treatment on the right.

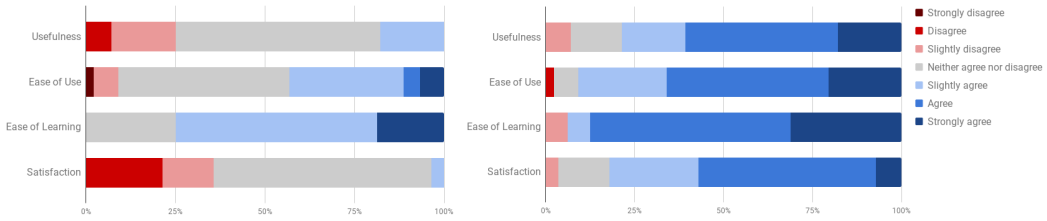


Figure 4 Results of the usefulness, ease of use, ease of learning, and satisfaction questionnaire of Asura: the control group (on the left) and the treatment group (on the right).

The results obtained from the questionnaire reveal improvements in the four metrics: usefulness, ease of use, ease of learning, and satisfaction. For instance, the usefulness of the control group had an average rating of 3.86 whereas the treatment group was 5.50 and the satisfaction valued 3.46 against 5.43, on the control and treatment groups respectively. From the free-text questions, it is possible to identify the main reason for these differences as being the feedback quality since some students in the control group pointed out feedback as a weakness (e.g., "Observations and program input/output could be improved" and "Feedback messages are scarce") while learners in treatment enjoyed the graphical feedback and let suggestions to further improve it in the chosen game (e.g., "On the game map, insert the name of the player above the ship" and "Insert a board that displays the remaining energy of the ship"). However, students have complained about the difficulty of getting started with the game and ES6 at the same time (e.g., "The idea is good, but when it is used to learn JavaScript since the start, it can be confusing because you have to learn JavaScript, learn the game, and think about the two."). The user interface of Enki has been criticized by students of both groups either because it lacks some features that they were expecting to have (e.g., terminal and debugger) or it looks bad on some devices (e.g., MacBook Air 13"). In regards to strengths, students emphasized the possibility of learning JavaScript through a game like the War of Asteroids (e.g., "Basic game to learn JavaScript, easy to get used to and easy to program against too.").

The analysis of the usage data aims to estimate the real efficacy of Asura in increasing practice time and, possibly, its impact on knowledge acquisition and retention. It is based on 6 variables, including the group (either control or treatment), number of validations, number of submissions, number of different days in which students have made an attempt (also known as return days), ratio of solved exercises in the course (a number between 0 and 1), and score obtained in the exam (an integer between 0 and 3), and a calculated attribute defined by the sum of the number of submissions and the number of validations, the number of attempts. The comparisons of the number of attempts, return days, the ratio of solved exercises, and exam grades between both groups demonstrate improvements in each of these quantitative metrics, as depicted in the boxplots of Figure 5.

The treatment group made 370 submissions of which 65 have been accepted whereas the control group submitted 44 times of which 16 succeeded. Adding this information to the percentage of exercises that were solved and the return days in both groups, it can be concluded that the students found it difficult to solve both kinds of challenges, but they struggled more in the treatment group than in the control group to overcome their difficulties. Furthermore, students in control only submitted until they solved the exercises while in the treatment they have made 40+ submissions than necessary, all of them in Chapter IV (i.e., the Chapter where competition starts). The amount of validations also reflects these trends,

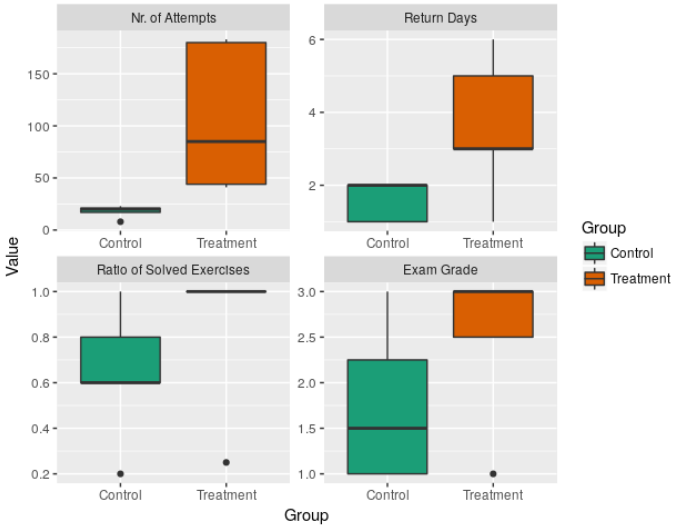


Figure 5 Quantitative comparison of metrics per group: number of attempts, return days, ratio of solved exercises, and exam grade.

yet the analysis suggests that students did not correctly understand the concept of Asura validations (i.e., friendly matches against the opponents) because they only validated 15 times in Chapter IV. The exam, which has been realized only by 4 learners of each group, shows that 75% of the students in treatment have achieved a good grade (between 2 and 3) and 50% in control. Nevertheless, the differences are so expressive due to the low number of participants that allowed other factors to be highlighted in the analysis.

5 Conclusions

This paper presents Asura, an automated assessment environment for game-based coding challenges, that aims to foster students' motivation requiring teachers a similar effort to that of creating traditional programming exercises. This environment offers teachers a framework and a CLI to author game-based programming challenges in which learners code an SA to play a game. These challenges introduce a competitive element in the form of tournaments similar to those realized in traditional games and sports, to further enhance the endeavor to develop better solutions that can beat their opponents.

The analysis of the data collected during the validation demonstrated relative success of Asura in accomplishing its goals. Nevertheless, results are based on a very small sample of students and, thus, insufficient to draw effective conclusions. The majority of the students of the control group expressed dissatisfaction with the amount and quality of feedback provided, whereas students of the treatment group have complained about the difficulty of getting started with the War of Asteroids, while also learning new concepts of ES6. Both groups pointed out a few issues in the user interface of the underlying system, Enki.

The next step is to enrich a repository of challenges already created, as from previous experience, instructors typically prefer to compile a set of existing exercises about a concept and use them.

References

1. Kirsti M. Ala-Mutka. A survey of automated assessment approaches for programming assignments. *Computer Science Education*, 15(2):83–102, 2005. doi:10.1080/08993400500150747.
2. Theresa Beaubouef and John Mason. Why the high attrition rate for computer science students: Some thoughts and observations. *SIGCSE Bull.*, 37(2):103–106, June 2005. doi:10.1145/1083431.1083474.
3. Esmail Bonakdarian and Laurie White. Robocode throughout the curriculum. *J. Comput. Sci. Coll.*, 19(3):311–313, January 2004.
4. Jens Dietrich, Johannes Tandler, Li Sui, and Manfred Meyer. The primegame revolutions: A cloud-based collaborative environment for teaching introductory programming. In *Proceedings of the ASWEC 2015 24th Australasian Software Engineering Conference*, ASWEC ' 15 Vol. II, pages 8–12, New York, NY, USA, 2015. ACM. doi:10.1145/2811681.2811683.
5. Stephen H. Edwards. Improving student performance by evaluating how well students test their own programs. *J. Educ. Resour. Comput.*, 3(3):1–es, September 2003. doi:10.1145/1029994.1029995.
6. E. Enström, G. Kreitz, F. Niemelä, P. Söderman, and V. Kann. Five years with kattis — using an automated assessment system in teaching. In *2011 Frontiers in Education Conference (FIE)*, pages T3J–1–T3J–6, October 2011. doi:10.1109/FIE.2011.6142931.
7. Ken Hartness. Robocode: Using Games to Teach Artificial Intelligence. *J. Comput. Sci. Coll.*, 19(4):287–291, April 2004. URL: http://dl.acm.org/citation.cfm?id=1050231.1050275.
8. IBM Community. Coderally, 2013. Last checked on November 2019. URL: https://ibm.com/developerworks/community/blogs/code-rally.
9. José Paulo Leal and Fernando Silva. Mooshak: A Web-based multi-site programming contest system. *Software: Practice and Experience*, 33(6):567–581, 2003. doi:10.1002/spe.522.
10. Arnold M. Lund. Measuring usability with the use questionnaire. *Usability interface*, 8(2):3–6, 2001.
11. José Carlos Paiva, José Paulo Leal, and Ricardo Queirós. Authoring game-based programming challenges to improve students' motivation. In Michael E. Auer and Thrasyvoulos Tsiatsos, editors, *The Challenges of the Digital Transformation in Education*, pages 602–613, Cham, 2020. Springer International Publishing.
12. José Carlos Paiva, José Paulo Leal, and Ricardo Alexandre Queirós. Enki: A pedagogical services aggregator for learning programming languages. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, pages 332–337. ACM, 2016. doi:10.1145/2899415.2899441.

A New and Interactive Teaching Approach with Gamification for Motivating Students in Computer Science Classrooms

Filipe Portela
Algoritmi Research Centre, University of Minho, Braga, Portugal
IOTech – Innovation on Technology, Porto, Portugal
cfp@dsi.uminho.pt

Abstract

Higher Education professors and students recognise that the introduction of new tools and learning methods can improve the teaching and motivation to learn. A new interactive and motivating methodology was designed and tested in a real classroom environment. This method, named TechTeach, explored a set of trending concepts applied to teach of Computer Science subjects: BYOD, Gamification, Soft-skills, quiz, and surveys and flipped classrooms to proportionate the best learning environment to the students. The paper presents the teaching plan and the case study used as proof of concept. In the end, it is possible to affirm that the students liked this method and are familiarised with it – most of the answers to the assessment method quiz (87%), was positive.

2012 ACM Subject Classification Social and professional topics → Information systems education; Social and professional topics → Computing education

Keywords and phrases Classrooms, Teaching, Soft-skills, Higher Education, Computer Science, Interac-tive approaches, BYOD, Flipped Classrooms

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.19

Acknowledgements I want to thank IOTECH for supporting the project.

1 Introduction

Nowadays, there is a massive difficulty of professors to motivate the students to their classes. Every year new challenges arise (new courses, students and coding languages to teach). Professors cannot domain all the concepts and technologies. So, it is time to change the paradigm. Professors cannot be a person who teaches but someone who explores new trends, ideas, concepts and motivate the students to learn. The use of flexibility, technology and innovation during the teaching process can lead to challenging learning environments [10] and highly motivating. According to [1] "Education, as it is, based on a model of skills, constitutes the development of utilitarian, stratified knowledge, that overvalues preparation for the labour market overtraining for the employment world, in its ontological value". The learning environments are transformed into flexible spaces that can be locat-ed within or outside the institution [10]. Recent technological developments have given rise to blended learning classrooms [10] that can be motivated by the use of Gamification. Gamification has generated increased attention recently across a range of con-texts [4] as is, for example, education. One of the most relevant changes occurred with the implementation of the Bologna model [1]. The main adjustment went through the development and the acquisition of general and specific skills, according to what professional profiles in the labour market determine [1]. In fact, the way of teaching is changing, and the students are less understanding and supporters of the old school. It is time to join a set of ideas and define a new approach to teach in computer science areas.

This paper is presenting a new way of teaching in Higher Education. This approach can combine a set of concepts: BYOD, flipped and inter-active classes, interactive quizzes and surveys, soft skills, with a focus in active learning.

The goal of this approach is increasing the interest and participation of students in classrooms by turning it more attractive and interactive. During a semester, this new approach was explored at the University of Minho, and the results are challenging and motivating (87% of positive answers and more than 85% of participation).

This paper is divided into six sections. After a brief introduction, the background presents the main topics of the work. Then, the methodology and the respective case study are presented. After this, results are discussed before the paper being ended at the conclusion.

2 Background

The approach presented in this article involves a set of concepts that it is relevant to explain.

2.1 Bologna Process

The Bologna Process was signed in 1999 [1] and became a reality of the European educational setting [9]. The objective of this declaration is to create a teaching system easily readable and comparable degrees by allowing promotion of the European dimension in higher education [9]. It is time to follow European directrices and pro-mote a teach from the qualification speech to the model of skills address [1].

2.2 Bring Your Own Device

According Moreira et al. [7] Bring Your Own Device (BYOD) is a subset of the consumerisation of Information Technologies (IT) as private or personally owned IT resources, such as computer device or software that are used for business proposes. In the case of education, BYOD consists of bringing laptops, smartphones or other devices to the classroom in order to increase active learning. Unfortunately, most universities still deliver instruction based on the philosophy of a teacher-controlled learning model that promotes passive learning [3]. This paper has the finality to show different ways of fostering active learners.Using this concept at the classrooms a set of interesting indicators can be collected (e.g. study the impact of the system access with the final grade) [7].

2.3 Flipped learning

Flipped learning is recognised as being an emerging instructional approach that can be used to support the pedagogy of teaching [3]. Learning environments can be any space when the students can learn and not only a classroom where learning is promoted [10]. A flipped classroom consists of using technology to push lectures outside of the school. The learning activities will be used to practice the concepts inside the classroom [12]. In the traditional classes, a lecturer exposes the topics, and then, the students have to do homework activities. In the flipped environment, students need to study and prepare the lesson. They will practice the contents of the week in-classroom activities with the professor and colleagues.

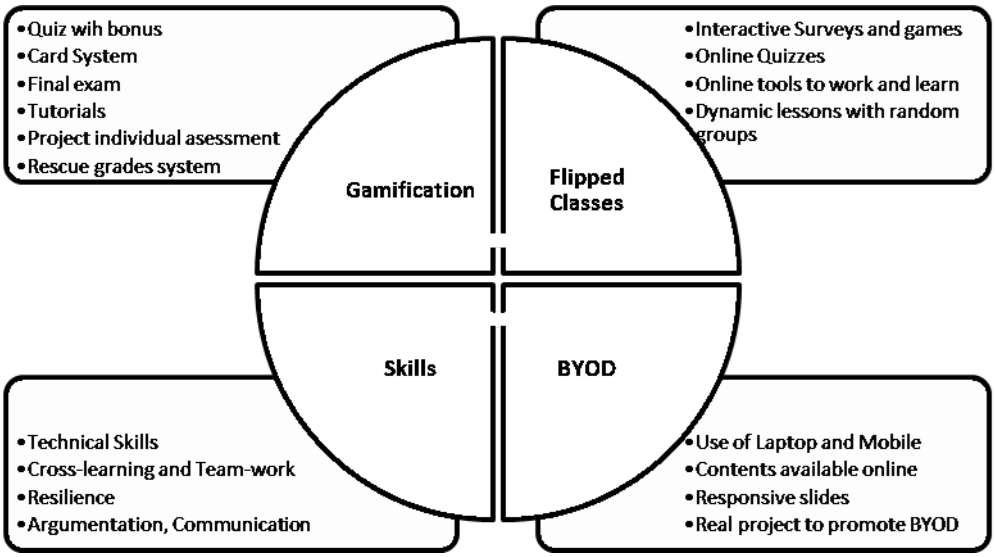


Figure 1 Main concepts of the method.

2.4 Gamification

Gamification consists of "using game-based mechanics, aesthetics and game think-ing to engage people, motivate action, promote learning, and solve problems" [6]. A past study [3] shows that there are several vital points which guide the deployment of the online gamified learning intervention. One research [3] demonstrated that online gamified learning activities have a positive impact on learning outcome.

3 Methodology

The methodology is based on a few numbers of concepts (BYOD, Soft-Skills, Flipped Classes and Gamification) and a set of methods/tasks, as can be observed in Fig. 1 and it is designed to courses with 10 ECTs with theoretical and practical classes.

The following list presents a brief overview of the methodology.

1. Theoretical classes are inverted and should be used to do a brief explanation of the topics and to do practical exercises
   a. Students must bring their laptop or smartphone to participate in in-class activities.
   b. Professors are encouraged to promote team-coding exercises.
      i. These exercises should be executed in a group of 3 students.
      ii. In each lesson, the groups must be different, and the active programmer must change.
   c. Each class should have a different learning challenge
   d. After the classes, the students should fill a quiz (multiple-choice) to assess their knowledge on some of the addressed topics.
2. Practical classes are used to develop a realistic project and stimulate soft-skills
   a. Each project should address a real problem of society and promote healthy competition between students.
   b. Projects should be divided into teams and if possible, groups
      i. A team is composed of a set of groups;
      ii. Each group should have different roles;
      iii. The groups are responsible for implementing a set of features

   c. Project meetings should cross different areas, students and knowledge. The sessions at classrooms should be divided
      i. By project features (groups)
      ii. By project roles (teams)
   d. During the class, the professor should analyse the work done by a student and evaluate their contribution to the project using a gamification system.
   e. Projects should have three assessment points:
      i. CP1 - to verify requirements and motivate the students, in a range of three results (10, 15, 20).
      ii. CP2 - to assess the technical quality of the project.
      iii. CPF - To assess the final result and the commercial potential.
   f. The project must include an anonymous peer evaluation using an N+1 scale. Each student should have the possibility to evaluate the contribution of each teammate for the outcome and to propose a project grade.
3. The professor is the "referee and manager" of the class ("game"), he should:
   a. Promote the team learning and the content research – Give some paths and cheats to the result and not provide the final answer.
   b. Give support to students when they require it and when it is under point a.
   c. Promote exercises comprising learning of soft skills (resilience, teamwork, public speaking, argumentation, work with uncertain, others).
   d. Create a list of Frequently Asked Questions (FAQs) with the most common issues verified by the students.
   e. Create a weekly quiz to assess the students' knowledge and implement a bonus system able to motivate the participation of students at the classes.
   f. Display videos able to show what is possible to do after concluding the course/subject. The videos also should explain some area trends and prognostics in the timestamp of five and ten years. Both contents should show the students what they can do in the real world after concluding the course. It motivates the student to participate in the subject.
   g. Turn available online presentations, videos, documents, practical examples and other essential contents.
   h. Promote a continuous assessment of the subject and show that the students' opinion is relevant.
   i. Implement and define the rules of the rescue system.
   j. Create Kahoots and games able to promote interactive discussion inside of the classroom.
4. Students are active learner. He is the leading "player" and should
   a. Study the topics before the lesson.
   b. Explore and learn new concepts
   c. participate in the "game", interact with the environment and train their soft-skills.
   d. Win points to achieve better grade possible.
   e. Contribute for the cross-learning and improvement and assessment of the CUnit.

A critical point of this article is not to show the methodology but explain how it can be applied in a real context.

4 Case Study

The methodology presented in section 3 was tested at the University of Minho during the first semester of 2019/2020 in the course unit (CUnit) of Web Programming (WP). This CUnit has more than one hundred (100) students, ten (10) ECTs and occurs during 20 weeks with 15 weeks of contact. Weekly, each student has the following hours:

- Theoretical (T): 2.
- Theoretical-practice (TP): 2.
- Laboratory (LP): 2.
- Non-presential: 7.

4.1 Week plan

The following list presents the most relevant tasks of CUnit plan grouped by weeks:

1stweek – presentation of the CUnit and implementation of a quiz to understand the class environment and students' profiles.

1. **T**: A Kahoot quiz was used to:
   a. Know student's opinion about the type of CUnit (Inverted or Normal). The answer compromises the student with the process.
   b. Understand the student's expectations and their situation in the class.
2. **TP**: Videos about the future of web programming are used to motivate the students.

2ndweek – flipped lessons started (in **T** lessons)

1. A set of exercises is proposed by class.
2. Students are invited to seat in different places to ensure a group of three random members.
3. During the class, the professor goes to each group, explaining some parts of the code.
4. When some critical issue is detected, the professor interrupts the exercises and explains it to everyone.

3rdweek – the project is presented (in **TP** lessons).

1. The project is about to create a system capable of supporting the development of outdoor activities (e.g. karting, rafting, orientation, others).
2. A set of thematic was presented, and each team has chosen one of them. The project is divided into three packages of features:
   a. Administration of the Outdoor Activity company.
   b. Mobile App to the participants.
   c. Management of the spaces company and sponsors activities.
3. A group of students develops each package. Every group has three areas: Front-end, Back-end and full-stack.
4. Each team has to prepare a contract document to delivery to the professors containing the project requirements and its cost (the final grade that they desire).

4^{th}week – the strategy of practical classes is defined.

1. **TP**: All teams work grouped by project roles; for example, all the full-stacks worked together.
   a. The full-stacks are responsible for ensuring the correct development of the project and connecting Front-end and Back-end.
   b. A set of roles are defined: Product Owner (team leader), Group Manager (one for each package) and Area Manager (one for each area).
   c. Students are motivated to define a week plan and share their experiences and difficulties during the group development.
2. **LP**: Each team works divided into groups in order to develop the respective features (packages).
   a. Students share the decisions taken, and tasks defined at the roles' meetings early occurred in LP classes.
   b. The development follows the rules defined by the team during LP classes.

5^{th}week – Quiz is launched.

1. A quiz about the topics discussed in each T is available to students answer after the class.
2. A bonus system is implemented
   a. The quiz is available to all the students that meet the T class.
   b. At each class, a set of students (between 5 and 15), is randomly selected to have the bonus.
3. Each quiz is composed of a set of questions with a limit of 100 points. Selected students have their result doubled (in case of 75 points, they receive 150).

7^{th}week – the yellow and red card system is implemented.

1. **LP**: The participation of each student at the practical component is evaluated using a gamification card system.
   a. A student can receive until two yellow cards. After that, they receive a red card and are reproved at the practical component.
   b. This system is used as an alert system for the students. They can know that they are not working enough, and if the student goes one like this, they will reprove at the CUnit. Otherwise, they receive the alert and improve their work.
2. The professor of laboratory classes starts the analysis of the project and can surprise the students by chosen someone to show what he did until the moment.
3. Professors ask the students about the work done, and, in case of the work done is none or too reduced, they admonished the student with a yellow card.
4. **T**: During the class, the professor shows the current probability of having a final exam. In the same lesson, he used a survey to collect the students' opinions about the CUnit performance and expectations until the moment.

10^{th}week – Professor asked students about their opinion (2^{nd} round).

1. **T**: Several questions were made regarding the CUnit: methods, professors and classes.
2. This Kahoot survey is essential to understand the student's opinion during class.
3. Students can rescue the grade achieved in the handwriting test

11^{th}week – Handwrite test

1. Students show what they know or learned
2. This test is individual and wants to test the basis of front-end and Back-end
3. There is no syntax validation; only the concept and idea are tested. In real-world, they can use anything to help them; however, they need to know how to start.
4. This test is used as a cut-off (binary result), i.e., some students are ready to continue, others not.

12^{th} week – Rescue system is activated

1. Students who were surprised by the Handwrite test and think that he knows more than the grade can show, they can rescue the MT classification.
2. The rescue system can maintain the student in the "game"; however, he needs to show more than the others. In this system, a particular focus is put in those students. Then, in case of success at the end, the final grade of MT is multiplied by 90%.

15^{th}week – A Game Group was developed recurring to Kahoot (T lesson).

1. All groups competed in order to be the best team.
2. The game is composed of 20 questions about the subject lectured. In the end, the students of the three best groups receive a bonus in the participation grade.

During the classes, students faced out some type of soft-skills challenge. For example,

(a) They had to work with different colleagues every week at T classes;
(b) Professors did not say all the answer but some part only. Students were encouraged to work with the uncertainty and look for solutions in internet, slides or books;
(c) TP classes are distributed by team roles (back-end, front-end and full-stack)
(d) LP classes are organized by group and project features.
(e) Project work (team and individual) are evaluated by all members of the group using a peer assessment tool (available at ioEduc).

After the method being introduced, the tasks and jobs continue in the following weeks. Next section presents the weeks with assessment points.

4.2 Assessment points

The control points of the project occurred for three weeks: eight (**CP1**), twelve (**CP2**) and seventeen (**CPF**). **CP2** and **CPF** had an individual and peer assessment.

Each student submitted their opinion about the grade of the group and the performance of each student. The degree of each group member varies from n-4 until n+4. The sum of all notes needs to be zero. For example, a group with a project of 12 can have students with 8 (-4) and others with 16 ($+4$). In case of a student did not work or worked less than 25% their colleagues can signalise him. The work of signalised students is then analysing by the professor and can be converted into a red card.

In CP2 and CPF, the professor can attribute yellow and red cards. A direct red card can be assigned in the case of a student being incapable of proving that they worked in the project or justify why they did not work. After CP2, the working plan is adjusted according to the remaining members.

Individual knowledge of each student is assessed through three mini-tests (MT) were designed. Each one was designed to evaluate:

(a) Front-end matters (Moodle test with a pool of questions).
(b) The basis of front-end and Back-end (handwrite code test without having consult and syntax validation).
(c) The entire content of CUnit (Moodle test with a pool of questions).

An algorithm was created to find the possibility of the students having an exam (percentage from 0 to 100) at the end of the semester. This algorithm used a Likert Scale [11] from 1 to 5 and took in the attention of six aspects:
(a) The motivation of the students (Positive)
(b) Preparation to the classroom (Positive)
(c) Noise during the lesson (Negative)
(d) Fatigue of the professor at the end of class (Negative)
(e) Meet of Class Goals (Positive)
(f) Hoarseness (Negative)

This algorithm is calculated at the end of each T class. Then, the students can know the probability of having an exam in three weeks: 6, 10 and 14. After the fourteen-week, students will see the final decision. In case of the percentage be upper to 50% exam will occur; otherwise, there is no exam.

4.3 BYOD Platform

A new tool named ioEduc[1] [8] was used to motivate interaction and learning. ioEduc is a Progressive Web APP (web/mobile platform) [5] designed to support teaching activities [7]. This platform was created by the author of this paper and then implemented by IOTech. ioEduc applies the concepts of Bring Your Own Device to classrooms and has a set of features / allow a set of tasks:
(a) Making student attendance at the classroom
(b) Taking notes of the lessons
(c) Rescue grade system
(d) Reading the slides (responsive system)
(e) Assessing the teammates work
(f) Creating teams and groups of projects
(g) Consulting the drive and the FAQ system
(h) Accessing to a real-time and offline chat (messaging system) with the professor.

For complementing the work, interactive classes are promoted using AWS C9[2] – "AWS Cloud9 is a cloud-based integrated development environment (IDE) that lets you write, run, and debug your code with just a browser [2].

Parallelly students were instigated to explore and deploy their project using
1. GitHub[3] – is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 40 million developers.
2. Heroku[4] – is a platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud.

[1] `https://ioeduc.iotech.pt`
[2] `https://aws.amazon.com/cloud9/`
[3] `https://github.com`
[4] `https://heroku.com`

Table 1 Description and goal of each class.

Type of Class	Description	Goal
T	Theoretical classes in groups of three students (random). Flipped classes. Discussion and analysis of the week topics. Exercising and practising examples.	Practice the concepts learned at home before class. Encourage group discussion and difficulty analysis; Share knowledge and experiences with different teammates.
TP	Students are grouped by team and area/role. The tasks of the project are defined.	Team working, test soft-skills, Promote the discussion, team learning and cross-learning
LP	Develop in a group, the tasks defined by the team. Monitoring of the project; Individual evaluation of the work (IEW). Support and monitoring the development of group projects, including feedback on their status;	Control Point - Monitoring and Evaluation of Project Status Motivate team working. Identify the students who are and are not working according to the rules.
Non-Presential	Reading, study and analysis of slides and CUnit book. Systematization of the concepts, principles and methods presented. Preparation for the next lectures. Development of a group project. Participate in the quizzes of topics.	Explore the capability of self-learning and studying something new. Assess the students' knowledge.

One of the most substantial aspects of this CUnit it the professors' accessibility. During the entire CUnit professors were available to help students after the classes by email. A chat is accessible in ioEduc to facilitate the communication between students and professor among the class.

4.4 Quality assessment of the UC

All the students are invited to evaluate the CUnit and participate in the definition of CUnit during the classes. They are asked to participate in interactive surveys (Kahoot) by answering questions about the performance of Professors, Type of Classes, Motivation, Expectations, among others. The assessment surveys are performed at the begin (1^{st} week), middle (6^{th} and 10^{th} week) and end (15^{th} week) of the course.

5 Discussion & Results

To a better comprehension of the CUnit plan, 1 presents the description and goal of each type of class (T, TP, LP).

The CUnit was assessed recurring three methods: Participation, Theoretical and Practice. Participation is assessed by the results achieved with the quizzes. The Mini-Tests evaluate the theory, and practice is measured through the project. Each method has a percentage associated. Participation has a particularity that it is essential to mention, the best grade without bonus has the maximum degree. For example, if the higher number of points without bonus is eight-hundred and fifty (850), this student will have twenty (20). All quizzes (with a

Table 2 Learning and assessment methods.

Method	Group	Goal
Surveys / Kahoots	BYOD Flipped Classes	Assess CUnit performance Ask students about their opinion Promote games and interactive discussions during the class
Card System	Gamification	Alert the students about their performance
Quiz	Flipped Classes	Assess assimilation of week concepts
Bonus	Gamification	Motivate students to participate in classes.
Project	Skills	Assess technical and soft-skills
FAQ	Flipped Classes	Help the students with the most common questions
Handwrite test	Knowledge Skills	Assess the expertise of doing the basis without help and syntax validation
Drive	Flipped Classes	Help the learning process with white papers, tutorials and examples
Rescue system	Gamification	The possibility of rescue a grade when the students think that he deserves more.
Game	Gamification BYOD, Skills	Play in the group, be fast, assess team knowledge and win points
Challenges with random groups	Flipped Classes BYOD, Skills	Promote the discussion and team learning
Final exam	Flipped Classes Gamification	The existence of the final exam is the responsibility of the students.

bonus) having a result higher than eight-hundred and fifty also has twenty, and all the other students have their grade in the percentage of 850. In this phase, it is essential to know the methods used to turn this subject more attractive and interactive.

In Table 2, it is possible to see the methods used and the goal of each method. For example, Yellow Card System was used to alert the student about their performance in the project. FAQ and drive were used to complement the teaching and give some tips and tutorials to students.

The methodology implemented in the Web Programming class was assessed. A survey using Kahoot and containing several questions was presented to the students in the last week (15).

Table 3 highlight the most relevant topics regarding the assessment methods. The survey was answered by ninety-three students (93), and the answers range from:

(a) negative | weak.

(b) neutral | acceptable.

(c) good | interesting.

(d) true Positive | excellent.

As can be observed in Table 3, most of the answers were positive (87% of the responses had 3 or 4 points). Regarding assessment methods, several tools can be explored.

Table 3 Final survey answer.

Question	1	2	3	4
Adequacy of strategies and methodologies adopted by the teacher	2	12	40	25
Work environment created	0	6	44	32
UC Global Appreciation – Theoretical	1	11	44	24
Overall, I appreciate UC	1	5	30	34

6 Conclusion & Future Work

Reflecting the transformations associated with the Bologna Process, it had worldwide proportions and has been raising various opinions [1]. It changed the way of teaching, and new strategies were defined. Besides that, the world is growing fast, and professors need to be ready for those changes. It is essential to invest in new ways of motivating students and promote the training of soft-skills.

Regarding the case study presented in this paper, a set of soft-skills was successfully trained:

(a) Problem Solving
(b) Decision Making
(c) Responsibility
(d) Cross-Learning
(e) Positive Attitude
(f) Resilience
(g) Team Working
(h) Communication
(i) Negotiation
(j) Reflection & Clarification
(k) Influencing
(l) Commitment
(m) Dealing with Aggression
(n) Stress Management
(o) Listening Skills
(p) Counselling Skills
(q) Presenting

New technologies and methods were implemented to motivate students and promoting continuous and active learning:

(i) Gamification used to drive participation, evaluate students' involvement in the classes and their intervention on the project.
(ii) Project and flipped classes used to improve skills.
(iii) BYOD was put in practice using a PWA named ioEduc.
(iv) Continuous assessment of the CUnit performed by the students.
(v) Rescue system available to students contest the grade
(vi) Hand-write test used as a cut-off system to assess the student has minimum knowledge required.

This paper showed new approaches that can be explored in computer science sclassrooms. Presented approach wants to motivate professors to explore different strategies to create active learners instead of following a traditional method. The scientific community should look to 2 as some examples of what can be done and take ideas to their classes. Professors should be confident and believe it works, and students will like.

This methodology revealed to be a success. The percentage of attendances at classes was around 85%, and 87% of the answers provided in the last quiz were classified as good or excellent. Achieved results demonstrate the students' interest and propensity to this type of classes.

In terms of digital lessons and online learning, this method will also be improved to consider non-presential classes. Although this new situation promoted by COVID19 brings new challenges, TechTeach can be easily adapted to a different type of lesson (synchronous or asynchronous). You can use, for example, ioEduc to share the slides online and online meeting tools (e.g. zoom, collaborate, team, among others,) to going along with the working group. You can also use Kahoot and ioEduc to provide the assessment tests. The extended version of this paper will explain how you can transit your teaching activity to a non-presential environment.

In the future, new mechanisms will be implemented like white and blue cards, new gamification process or new methods of theoretical assessment. In the sequence of this paper, an extended version will be published. The extended version will have a depth analysis of the student's opinion. Regarding to digal lessons and online learning, this method will also be improved in order to consider non-presential classes

References

1 C.V. Araújo. The bologna process and curricular changes at higher education: what are skills for? *Educação e Pesquisa*, 44, 2018.

2 A.W.S. Aws cloud9, 2020. Retrieved 1 11, 2020, from. URL: `https://aws.amazon.com/cloud9/`.

3 C. Davis. Flipped or inverted learning: Strategies for course design. In *Enhancing Instruction with Visual Media: Utilizing Video and Lecture Capture*, page 25. Information Science Reference, 2013.

4 P.B. Doyle. Gamification and student motivation. *Interactive Learning Environments*, 24(6):1162–1175, 2016.

5 Filipe Portela Gisela Fernandes and Manuel Filipe Santos. Pwa and pervasive information system - a new era. In *Advances in Intelligent Systems and Computing (WorldCist 2020 - PIS Workshop)*, Portugal, 2020. Springer.

6 K.M. Kapp. . *The gamification of learning and instruction: Game-based methods and strategies for training and education.* John Wiley & Sons, 2012.

7 A. Maia, F. Portela, and M. F. Santos. Web intelligence in higher education: A study on the usage of business intelligence techniques in education. In *2018 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pages 176–181, 2018.

8 Diogo Ferreira Miguel Silva and Filipe Portela. ioeduc - bring your own device to the classroom. In *Conference: ICPEC 2020 - International Computer Programming Education Conference*, Portugal, 2020. OASIcs.

9 C. Partenie. The bologna process: Between past reforms and the innovative future. In *Conference: International Multidisciplinary Scientific Conferences on Social Sciences and Arts*, Bulgaria, 2014. Albena.

10 M.S. Ramírez-Montoya. Inverted learning environments with technology, innovation and flexibility: Student experiences and meanings. *Journal of Information Technology Research (JITR*, 9(1):18–33, 2016.

11 John Robinson. Likert scale. In *Encyclopedia of Quality of Life and Well-Being Research.* Splinger, 2014.

12 J.F. Strayer. How learning in an inverted classroom influences cooperation, innovation and task orientation. In *Learning Environments Research*, page 16. Springer, 2012.

Gamification of Learning Scratch in Elementary School

Serhii D. Prykhodchenko
Department of Software Engineering, Dnipro University of Technology, Ukraine
prykhodchenko.s.d@nmu.one

Oksana Yu. Prykhodchenko
Department of Finances, National Metallurgical Academy of Ukraine, Dnipro, Ukraine
oksana.prykhodchenko@gmail.com

Olha S. Shevtsova
Department of Software Engineering, Dnipro University of Technology, Ukraine
shevtsova.o.s@nmu.one

Sergii Yu. Semenov
Department of Software Engineering, Dnipro University of Technology, Ukraine
semenovs@gmx.com

Abstract

The article deals with the problem of gamification for primary school students. The main idea of creating a software product was to create the correct programming sequences for solving simple programming problems in the Scratch programming environment. The technological preconditions for creating an application are described; a review of gamification basics has been carried out. In this article, we first illustrate the current art of state of gamification. Then we discuss requirements for teaching and studying Scratch for primary school students, and then we describe the development of a game application for teaching Scratch programming constructions. In the experimental section we compare the results of the experimental and control groups of primary school students, which showed differences in the levels of knowledge of primary school students after the experiment, where the results of the experimental group are higher than results of the control group

2012 ACM Subject Classification Social and professional topics → Computing education programs; Social and professional topics → Computing education; Social and professional topics → Informal education

Keywords and phrases Education, Gamification, Scratch, Game Mechanic, Programming Language

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.20

1 Background

The relevance of research: creating a simulator program in the form of games will help make learning process exciting and interesting. Competitions and rewards for achievement allow students to improve their status and get another form of expression but also get an incentive for persistence, creativity.

Gamification is a concept that means using of scenarios specific to computer games in computer tools, in areas far from gaming [6, 12, 13, 10]. However, the use of games does not replace traditional lessons. On the contrary, it provides an additional opportunity for learning. There are 7 trends among modern teaching technologies in elementary education: mLearning or mobile learning [9, 3]; Storytelling [16, 18]; Edutainment (education entertainment) [8, 17]; Microlearning [10, 4]; Blended learning [5]; STEM-projects (education) (science, technology, engineering, mathematics) and robotics, LEGO-construction [7, 15]; Gamification (e-learning) [6, 10, 8, 17, 4].

Gamification can be used in the following cases:

- to develop certain skills or behaviors [19, 1];
- to visualize and emphasize activities and skills that are difficult to demonstrate using traditional techniques [4, 19, 1];
- to interest students, to create a kind of competition between them [6, 12, 19];
- to monitor students' progress [19, 11].

Research aim: study gamification teaching influence on the quality of learning new material by elementary school students. Research question: Influence of gamification on the quality of learning in primary school programming lessons. Therefore, we have the objectives: to analyze and test effectiveness, efficiency, and possibility of applying modern direction in education – gamification, in the work of elementary classes teacher. Gamification methods have become widespread in the educational field. Educational computer game is a software tool that allows you to direct activities of a child to achieve a certain didactic goal in the form of play. It is not isolated from the pedagogical process and is offered alongside traditional games and training; it does not replace ordinary lessons, but complements them, enriching the pedagogical process with new opportunities. Most of educational computer games offer such elements of knowledge that under normal conditions are difficult to understand or learn [6, 12].

The computer world is always secondary, it has nothing that would not be contained in the real world or imagination of its creator, but at the same time it is not limited by frames of physical laws, it has any resources to recreate the situation, a virtual realization of most fantastic ideas.

An unusualness of an imaginary situation in a training computer game is that the player acts within its frames, but cannot change it. The computer imaginary situation is external to a child. A player does not create it but gets into it.

In computer reality, it is always possible to go back to tasks, replay, try other options. The pre-orientation stage in a computer-based educational game is not at the semantic level, but above all at the action level.

Rules of educational computer game exist before its beginning, may be disclosed, but are not generated during the game. Computer learning games and exercises should be considered as a special means of stimulating children's creative activity. They are interesting and accessible, and their game tasks contain motive and purpose, as well as ways and means of solving them. Scratch is an object-oriented visual programming environment that enables you to create computer animations, multimedia presentations, interactive stories, games, models, and more. Scratch is a freely distributed educational program that can be downloaded from the developer's official site (`https://scratch.mit.edu/`).

The programming is as follows: users "assemble" in "drag-and-drop" style the program from blocks that have objects and scenes. An object that is associated with a particular image, set of variables, and command blocks to determine its behavior is called a sprite. You can modify the sprite by importing it from the built-in library (categories include animals, fiction, letters, people, things, transport), or create using a built-in graphic editor or other software. Commands-blocks are grouped into certain groups: "Movement" (performing the movement of sprites), "View" (changing the sprite patterns, its text dialogues), "Sound" (sound commands, volume, tempo), "Pencil" (graphical construction) images), "Manage" (looping, branching), "Sensors" (information about touching objects and determining distances between them), "Operators" (performing mathematical and logical operations, selecting a random number),

"Variables" (creating variables, assigning them certain values). As a whole, Scratch can be described as easy to use and powerful enough to meet the challenge of creating your own programming for beginners.

In the curriculum of Ukrainian elementary schools, the subject "Informatics" is related to the study of computer technologies. This discipline is studied from the second grade, while under the curriculum, in the third grade, students begin to study the Scratch programming language. Scratch is not always clear to students in Ukraine, especially in the initial stages, as well as in another countries [14].

2 Designing the game logic

As we said before, the constructions of the Scratch language for elementary school students are not always clear. Thus, to better understanding the language constructions, it is proposed to create a game in which the student will create a Scratch structure in a game with prompts from the teacher or from the game itself.

We have set the following requirements for the upcoming game:

- it should be interesting, develop attention, speed of reactions, train memory;
- completing all play tasks should teach a child to think analytically in unusual situations, to classify and summarize concepts; develop fine motor skills and visual-motor coordination;
- the game should be thoughtful and simple at the same time, with low levels of aggression.

We have also developed rules of the game:

- the number of minutes per game is equal to a child's age multiplied by 1.5. For example, for an eight-year-old child, the game lasts 12 minutes.
- the number of sessions per game is a maximum of 3 per day.
- after work that is mandatory to have eye exercises and rolling games.

The initial purpose of the work was to make a learning game that would be understandable and interesting for the children.

The main goal is to test the game, to draw some conclusions about the students' interest in game learning technologies, to test students' knowledge, and to find out how the game influenced learning of the material.

The game has two modes – "Assembling without false targets" and "Assembling with false targets". The main character is a robosphere. The design was chosen in a way that it was not difficult for the children to adjust to the game. The main map is "located" in space (similar to the game "Ballance" (Fig. 1)).

The purpose of the game is to assemble an answer to a test question that is randomly selected from database. Movements of main actor is controlled using the WASD buttons and the arrow keys.

The database of Scratch expressions is formed; then it divided into single elements – keywords and their parameters. Each expression is associated with test question, and each element has its serial number in the expression. At the beginning of the round, elements of one of the random test expressions called "correct elements" are randomly placed on the game board and test question is displayed over board to the player.

There are two game modes:

- in the first, you need to collect the test expression in the correct order from the elements of only the test expression itself.
- In the second mode, not only "correct elements" are placed on the playing field, but also random elements from other expressions called "false targets".

Figure 1 Ballance Computer Game.

The player must move the robotic sphere to assemble in the correct order the “right elements” into a full Scratch expression.

Game algorithm as steps sequence:

1. Select a game mode.
2. Extract a test expression from the expression base
3. Divide it in order into the “correct” elements.
4. Store in memory the length of the expression.
5. Extract the test question associated with expression for the student.
6. Assign each of the “correct elements” its serial number in the expression.
7. Set each of the correct elements weight = 1.
8. If the second game mode is selected, then randomly add a random number of “false targets”, limited by the number of elements of the test expression.
9. Set for each “false target” serial number = -1 and weight = -0.5.
10. Distribute the “correct elements” and “false targets” as items randomly across the playing field.
11. Create an empty list for the “Correct Elements” sequence
12. Assign the last element the number 0. Assign the total points = 0.
13. Start the game.
14. Display the test question for the student on the screen.
15. Wait for student’s action.
16. If the student has collected the item, then
17. If the item serial number is 1 more than the last item in the list and weight = 1, then add this item to the list of “correct elements” as the last element and add weight to the total points.
    a. Otherwise, subtract the weight from the total points, and put the collected item on a random free space on the board.
18. Check if the length of the list is equal to the length of the test highlight, then end the game by going to step 19.
    a. Otherwise, go to point 14.
19. Display the test question again and the complete test expression for the student.

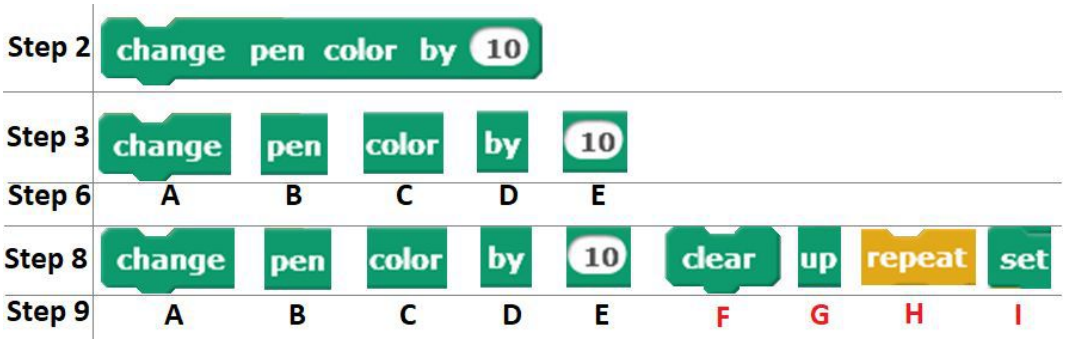


Figure 2 Illustrated steps of algorithm.

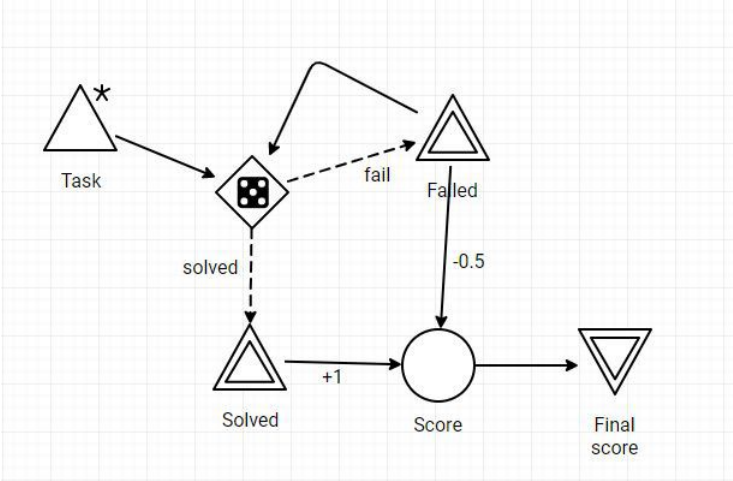


Figure 3 "Scratched ballance" rule model.

20. Display the final score. If it is equal to the length of the test expression, then congratulations on a complete victory
    a. If the sum of the points is greater than 0, but less than the length of the test expression, then congratulations on the success.
    b. If the score is less than 0, then congratulate player on test completion and wish him success and efforts in learning.

An example of the operation of some steps of the algorithm is presented in Fig. 2.

At the second step, one of the test expressions is retrieved from the database. At the third step, the expression is divided into separate "correct" elements. At the sixth step, each of the elements is assigned a serial number A = 1; B = 2; C = 3; D = 4; E = 5. In the eighth step, false targets are added. On the ninth, false targets are assigned serial numbers equal to "−1" so that F = −1; G = −1; H = −1; I = −1.

The game logic proposed in the developed game "Scratched Balance", based on "Gamification patterns for gamification applications" [2], and can be represented in the form of the scheme shown in Fig. 3 where Task is a result of Player's action, diamond – model for good/bad choice, Failed and Solved imitates rules for scoring. When the Score is full, so we get a final Score of the round.

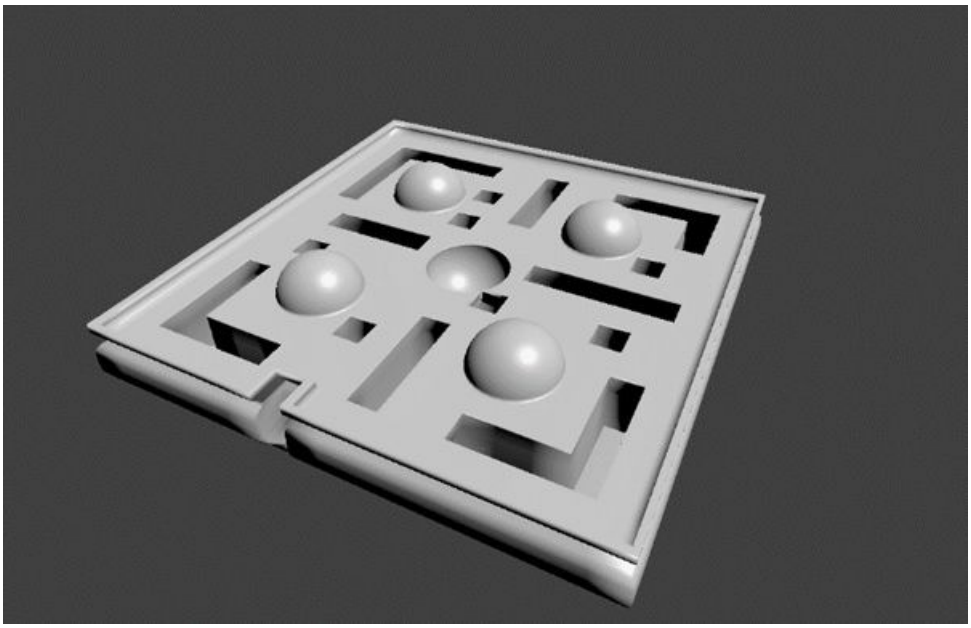


Figure 4 Model of the playing surface.

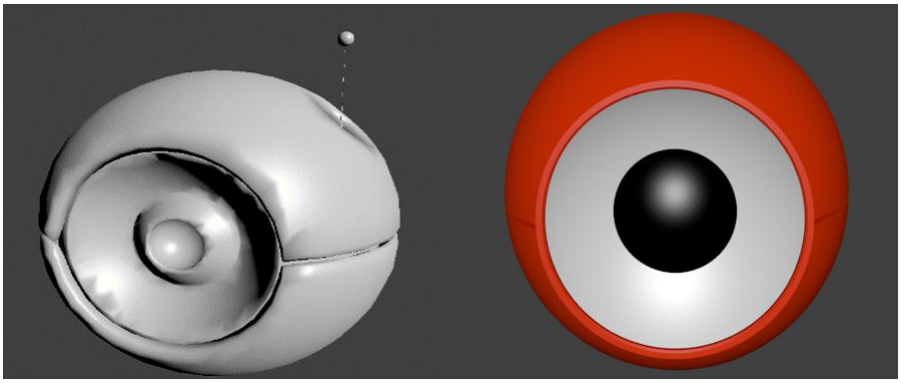


Figure 5 The main character – Robo-sphere.

2.1 Landscape of the Game

To increase the interest of children, elements of balancing the robosphere on uneven surfaces were introduced into the game, which makes it difficult to achieve the goals of the game. This raises the interest of children in the game. The difficult stage is to come up with an interesting and unusual design of the game board itself (Fig. 4).

Then, paint our model. To do this, we need to find textures and "overlay" them on the model. We chose the texture of wood and stone.

The main character of our game – Robo-sphere (Fig. 5).

For the player to be able to move around the terrain, we need to create a control unit – a script, which we will call "Roll". It will be responsible for moving the character around the area. In the main game function that is responsible for the game itself, we set a simple task to assemble the correct sequence of Scratch language operators capable of performing the simple operation asked in the "test question". After creating the interface, we got the following look at the field (Fig. 6).

Figure 6 Game board.

The main goal for the player is to create a simple structure in the Scratch language that will fulfil the set condition. For example: “set the property “colour” of the object “pen” to 5”; “Wait 10 seconds”, “move to position 10, 5”, etc. At the beginning of the game, the task is issued in capital letters for 5 seconds, after which the game begins. If the student has forgotten the task, or wants to read it again, he can call the task in a pop-up window by pressing the F1 key. The gameplay are series of the robosphere movements to collect the correct sequence of operators on the playing field. For example, for the test question “Assign the property “Colour” of the object “Pen” to 5”, the player must collect the sequence of operators “SET”, “PEN”, “COLOR”, “TO” and “5” (Fig. 6)

3 Experimental methodology

To study the impact of gamification on learning, we have selected students of grade 3 of school №15. The study was conducted in the 1st semester when the Scratch course has not been studied before. Knowledge quality control was performed through testing (each child was given a task consisting of 16 examples containing Scratch action). The experiment took place in two stages and lasted for three weeks.

In the first stage, the rules of simply Scratch operators were explained to the children and tested. After that, they had the opportunity to play “Scratched Balance”. The training game was installed on an Acer laptop that was connected to a 40-inch TV that was used as a display. This made it easy for all children to follow the play process. But according to our observations, most of the students watched the game directly through the laptop and tried to give the player immediate advice (Fig. 7). At the end of the experiment, we retested the same tasks.

Figure 7 Testing the game.

4 Results of the study

The students of the third grade of the school studying computer science were divided into two groups based on the classes in which they study: "3A" class was a control group and consisted of 24 students, "3B" class was the experimental group consisted of 26 students.

As a first step students of both classes were tested. First test consisted of 16 problems and student got 1 point for each, so 16 points were maximal result. The test results were processed using the t-test. According to the statistical results of preliminary testing, the average score of the experimental group was 4,615 points, and the average score of the control team – 4,625 points. Thus, we compare the value of the t-criterion ($t = 0.01416$) with the critical value at $p = 0.05$, which is 1.677. Since the calculated value of the criterion is less than the critical one, we conclude that the observed differences are statistically insignificant. These calculations showed that the students of the experimental and control groups had the same level of knowledge in computer science before the experiment. After three weeks we conducted the second test. Then we compared the results of the experimental and control groups after the learning. Test results of first and second tests (annex 1) were used for the both groups.

The table presents the results of the data processing after excluding the influence of covariance (test points before the experiment) on the test results after the experiment.

Table 1 The ANCOVA test results for learning achievement from post-test of the two groups.

Group	M	S.D.	Adjusted Mean	S.E.	N
Control group	7.25	2.56	7.24541	0.264455	24
Experimental group	11.88	2.55	11.88885	0.25408	26

Source: own

Since the calculated value of criterion $F = 160.32$ is more critical, we conclude that the observed differences are statistically significant ($p < 0.05$).

The mean score of students in the experimental group was 11.88, and the standard deviation was 2.55, the mean score in the control group was 7.25, and the standard deviation was 2.56.

These calculations showed that the students of the experimental and control groups had different levels of knowledge of computer science after the experiment.

5 Conclusions

The idea of the game was to introduce an element of the game into the study of standard Scratch language constructions to attract the interest of elementary school students. The game can be used by elementary school teachers as supporting material in the learning process of Scratch students, because the calculated value of criterion F = 160.32 is more critical and we conclude that the observed differences are statistically significant with $p<0.05$.

This research showed that Game-based learning have some influence on the performance of elementary school students. It's obvious that understanding the material in the discipline under consideration has increased when comparing student groups, which can serve as an indicator of the success of game application.

Based on the results of our work, we recommend introducing the elements of gamification into teaching the Scratch language in Ukrainian elementary schools, since our study showed a marked excess of the results of the experimental group on the results of the control group of students.

References

1 Raed S Alsawaier. The effect of gamification on motivation and engagement. *The International Journal of Information and Learning Technology*, 2018. doi:10.1108/IJILT-02-2017-0009.

2 Darius Ašeriškis and Robertas Damasevicius. Gamification patterns for gamification applications. *Procedia Computer Science*, 39:83–90, December 2014. doi:10.1016/j.procs.2014.11.013.

3 M. S. Bhullar. A new method of learning: M-learning (mobile learning). In *2014 9th International Conference on Computer Science Education*, pages 322–325, 2014. doi:10.1109/ICCSE.2014.6926478.

4 Laurie Butgereit. Gamifying mobile micro-learning for continuing education in a corporate it environment. In *2016 IST-Africa Week Conference*, pages 1–7. IEEE, 2016. doi:10.1109/ISTAFRICA.2016.7530597.

5 D Randy Garrison and Norman D Vaughan. *Blended learning in higher education: Framework, principles, and guidelines*. John Wiley & Sons, 2008. doi:10.1002/9781118269558.

6 Thomas Hainey, Thomas M. Connolly, Elizabeth A. Boyle, Amanda Wilson, and Aisya Razak. A systematic literature review of games-based learning empirical evidence in primary education. *Computers & Education*, 102:202–223, 2016. doi:10.1016/j.compedu.2016.09.001.

7 JP Holdren, C Marrett, and S Suresh. Federal science, technology, engineering, and mathematics (stem) education 5-year strategic plan. *National Science and Technology Council: Committee on STEM Education*, 2013.

8 U. Jayasinghe and A. Dharmaratne. Game based learning vs. gamification from the higher education students' perspective. In *Proceedings of 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, pages 683–688, 2013. doi:10.1109/TALE.2013.6654524.

9 Agah Tugrul Korucu and Ayse Alkan. Differences between m-learning (mobile learning) and e-learning, basic terminology and usage of m-learning in education. *Procedia - Social and Behavioral Sciences*, 15:1925–1930, 2011. 3rd World Conference on Educational Sciences - 2011. doi:10.1016/j.sbspro.2011.04.029.

10 A. Martens and W. Mueller. Gamification - a structured analysis. In *2016 IEEE 16th International Conference on Advanced Learning Technologies (ICALT)*, pages 138–142, 2016. doi:10.1007/978-981-4560-52-8_66-1.

11 A Matallaoui, N Hanner, and R Zarnekow. Gamification: Using game elements in serious contexts. *Springer, Cham, Switzerland*, 2016.

12 Manuel Ninaus, Kristian Kiili, Jake McMullen, and Korbinian Moeller. Assessing fraction knowledge by a digital game. *Computers in Human Behavior*, 70:197–206, 2017. doi:10.1016/j.chb.2017.01.004.

13 Manuel Ninaus, Korbinian Moeller, Jake McMullen, and Kristian Kiili. Acceptance of game-based learning and intrinsic motivation as predictors for learning success and flow experience. *International Journal of Serious Games*, 4(3), September 2017. doi:10.17083/ijsg.v4i3.176.

14 Ana Maria Ortiz-Colon and Jose Luis Maroto Romo. Teaching with scratch in compulsory secondary education. *International Journal of Emerging Technologies in Learning (iJET)*, 11(02):67–70, 2016. doi:10.3991/ijet.v11i02.5094.

15 Margarita Elizabeth Ortiz Rojas, Katherine Chiluiza, and Martin Valcke. Gamification in higher education and stem: A systematic review of literature. In *8th International Conference on Education and New Learning Technologies (EDULEARN)*, pages 6548–6558. Iated-int Assoc Technology Education A& Development, 2016. doi:10.21125/edulearn.2016.0422.

16 Panagiotis Psomos and Maria Kordaki. Pedagogical analysis of educational digital storytelling environments of the last five years. *Procedia - Social and Behavioral Sciences*, 46:1213–1218, 2012. 4th World Conference on Educational Sciences (WCES-2012) 02-05 February 2012 Barcelona, Spain. doi:10.1016/j.sbspro.2012.05.277.

17 Jihan Rabah, Robert Cassidy, and Robert Beauchemin. Gamification in education: Real benefits or edutainment? In *European Conference on e-Learning*, pages 489–XIX. Academic Conferences International Limited, 2018. doi:10.13140/RG.2.2.28673.56162.

18 Bernard Robin. The educational uses of digital storytelling. In Caroline M. Crawford, Roger Carlsen, Karen McFerrin, Jerry Price, Roberta Weber, and Dee Anna Willis, editors, *Proceedings of Society for Information Technology & Teacher Education International Conference 2006*, pages 709–716, Orlando, Florida, USA, March 2006. Association for the Advancement of Computing in Education (AACE). URL: https://www.learntechlib.org/p/22129.

19 Michael Sailer and Lisa Homner. The gamification of learning: A meta-analysis, 2019. doi:10.1007/s10648-019-09498-w.

A Results (number of correct answers) of every student

Table 2 Annex A. Results (number of correct answers) of every student.

№	Group	First test	Second test	Group	First test	Second test
1	Control group	1	3	Experimental group	8	14
2	Control group	2	4	Experimental group	2	8
3	Control group	4	5	Experimental group	8	13
4	Control group	1	3	Experimental group	7	12
5	Control group	5	8	Experimental group	1	6
6	Control group	2	6	Experimental group	3	10
7	Control group	6	9	Experimental group	7	14
8	Control group	7	9	Experimental group	6	15
9	Control group	4	5	Experimental group	4	10
10	Control group	3	6	Experimental group	8	15
11	Control group	5	9	Experimental group	3	10
12	Control group	8	11	Experimental group	2	10
13	Control group	7	9	Experimental group	7	15
14	Control group	4	8	Experimental group	8	15
15	Control group	7	9	Experimental group	2	10
16	Control group	8	12	Experimental group	5	12
17	Control group	7	10	Experimental group	3	11
18	Control group	2	6	Experimental group	1	9
19	Control group	3	6	Experimental group	2	9
20	Control group	5	8	Experimental group	7	15
21	Control group	7	10	Experimental group	4	11
22	Control group	6	9	Experimental group	5	13
23	Control group	6	4	Experimental group	3	12
24	Control group	1	5	Experimental group	2	12
25	—	—	—	Experimental group	8	16
26	—	—	—	Experimental group	4	12

Source: own

Computer Programming Education in Portuguese Universities

Ricardo Queirós
CRACS – INESC-Porto LA, Portugal
uniMAD, ESMAD, Polytechnic of Porto, Portugal
http://www.ricardoqueiros.com
ricardoqueiros@esmad.ipp.pt

Mário Pinto
uniMAD, ESMAD, Polytechnic of Porto, Portugal
mariopinto@esmad.ipp.pt

Teresa Terroso
uniMAD, ESMAD, Polytechnic of Porto, Portugal
teresaterroso@esmad.ipp.pt

Abstract

Computer programming plays a relevant role in the digital age as a key competency for project leverage and a driver of innovation for today's modern societies. Despite its importance, this domain is also well known for their higher learning failure rates. In this context, the study of how computer programming is taught is fundamental to clarify the teaching-learning process and to ensure the sharing of the best practices. This paper presents a survey on computer programming teaching in the first-year courses of Portuguese Universities, more precisely, what is taught and how it is taught. The study focuses essentially on the following facets: the class characterization, the methodologies used and the languages/technologies taught. Based on these criteria, a survey was done which gathers information of 59 courses included in a wide range of Universities spread across Portugal. The results were collected and analyzed. Based on this analysis a set of conclusions were taken revealing some interesting results on the teaching methods and languages used which can be useful to support a discussion on this subject and, consequently, to find new paths to shape the future of programming teaching.

2012 ACM Subject Classification Social and professional topics → Computer science education

Keywords and phrases computer programming, teaching-learning, universities

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.21

Funding This work is financed by National Funds through the Portuguese funding agency, FCT – Fundação para a Ciência e a Tecnologia, within project UIDB/50014/2020.

1 Introduction

Computer programming is considered one of the most important and emerging domains in today's society. As a domain with large market demand, educational institutions have been including in their curricula, a set of related disciplines, ranging from the introductory level to a more advanced one.

At the same time, this domain has high levels of failure, especially in introductory programming disciplines. There are several reasons for this fact [2, 5, 14, 11], ranging from traditional teaching methods, the difficulty of students in enhancing the problem-solving facet, the small and limited number of programming exercises, to the lack of automated tools to assist teachers in authoring and evaluating exercises and students in monitoring their resolutions [4, 6].

With the advent of automatic tools for evaluating programming exercises, which came in part to relief the teachers from the manual burden of manual assessing of student's code, and the Web learning environments that came to provide more sustained guidance to students while solving exercises, we continue to assist to a great lack of motivation in learning programming [1, 8].

In the last few years, in order to engage students and foster a collaborative and competitive spirit, elements associated with the mechanics that we typically see in games, the so-called gamification, began to emerge [12]. Gamification is now a crucial component in learning environments and has been used in order to motivate students to remain focused on overcoming their difficulties. Despite the huge buzz with gamification in education, the lack of systems that can be easily integrated into learning environments and their unbalanced use, reveals that this technique has not yet been fully exploited [7].

This paper presents the current state of the teaching-learning process of computer programming in Portugal. For this study, a survey was defined and distributed to several Portuguese educational institutions, more specifically, to teachers who teach introductory programming subjects. This survey raises several questions related to the characterization of the classes, the teaching methods, the languages and tools used, the reasons for the current difficulties and the desired improvements. Based on the results of the survey, an analysis is made that essentially aims:

- to characterize the teaching of computer programming in Portugal;
- to know which methodologies, languages and tools are adopted;
- to identify good practices implemented (with satisfactory results);
- to outline lines of action for the future of programming teaching.

The remainder of the paper is structured as follows: Section 2 explores the reasons for the failure of teaching programming. The next section describes the experience made to obtain data on teaching programming in Portugal, namely, the methodology used for data collection. The results analysis section presents the results of the survey and analyzes them. Finally, the conclusions, the main contributions of the article and possible paths for future work are presented.

2 Programming education issues

In this section, we begin by identifying the main reasons for the difficulties that teachers and students have in the teaching-learning process of computer programming.

In order to learn to program it is not enough to know the syntax of a language. There is a set of inherent concepts that requires a level of abstraction and structured reasoning from the student, which is difficult to achieve, especially in an introductory phase. Several scientific studies points out several reasons for failure in this area [2, 5, 14, 11]:

- Complex domain of complex programming;
- Traditional teaching and study methods;
- Psychological aspects;
- Difficulties in using/integrating automated tools.

In the next subsections, we explore all these facts.

2.1 Complex domain

Programming learning requires a range of skills ranging from problem-solving to abstraction. These skills associated with reasoning structured in order to find the best solutions for a given problem are decisive for successful progress in this domain. Several studies show [9, 11, 5], that, at an early stage, students have difficulties in assimilating all of these skills.

In fact, problem-solving is nowadays seen as one of the main soft-skills that anyone must have in order to be successful in their work. This skill is essentially characterized by five steps: In a **first phase**, you start by analyzing a problem (for instance, a programming assignment) and identifying what needs to be addressed to obtain a solution. In this step, the necessary skills focus on good reading, interpretation, and analysis of the problem and adequate identification of the requirements.

In a **second phase**, and after realizing the problem and identifying needs, it is time to discuss possible solutions. It is rare that a single strategy is an obvious answer to solving a complex problem. The creation of a set of alternatives helps to cover all needs and reduces the risk of exposure if the first strategy that implements fails. At this stage, the necessary skills focus on good planning for solving a problem (e.g., developing algorithms).

In a **third phase**, the best solutions are evaluated. Depending on the nature of the problem, the evaluation of the best solutions can be carried out taking into account several criteria (e.g. getting from point A to point B more quickly or spending less money). Here the necessary skills are discussion and teamwork, prioritization and test development.

In a **fourth phase**, the decision reached in the previous phase is implemented. Here, a programming language is typically used to implement the best solution in order to solve the problem. As necessary skills, we highlight the ability to codify and collaborate (in this case, group work).

Finally, the effectiveness of the solution execution is evaluated.

Many students have also deficits in mathematical and logical knowledge. Several experiments [9] were carried out to find correlations between mathematical knowledge and the lack of programming skills. In these experiments, the authors concluded that the students involved had profound difficulties in several areas, such as basic calculus and theory of numbers or simple geometric and trigonometric concepts. The authors also report difficulties related to the transformation of a textual problem into a mathematical formula that solves it. Limitations in terms of abstraction and logical reasoning have also been identified.

At the same time, and still related to the nature of programming, another problem persists that is closely linked to the syntax of languages. In fact, the syntax of languages is complex (in fact, they were designed to be used at a professional level and not to support your learning) and, in some cases, has evolved in a meteoric way [13], making students have difficulties in its adaptation, memorization and consequent application. Obviously, these problems can be alleviated by teaching and study techniques that are discussed in the next subsection.

2.2 Traditional teaching-learning methods

One of the main problems in teaching programming has to do with the fact that teachers are typically more focused on teaching a programming language and its syntax, rather than promoting problem-solving using a programming language. This enormous emphasis on syntax, at the expense of a more practical component, is an obstacle to student's progress. Also, in the programming area, where there is a great need for teaching dynamic concepts, this is usually done using materials of a static nature (e.g. drawings on the board, slides that are too long and confusing and verbal communication, sometimes deficient and of difficult understanding).

At the same time, the study methods are also not the most suitable. Programming requires a very practical and intensive study. Obviously there are many disciplines that require study methodologies based on reading and memorizing formulas or procedures. However, programming, like mathematics, requires a different method of study that involves intense training. The only way to learn to program is to program. Just watching classes, watching videos and reading specialized books is not enough. Moreover, students tend to give up problems whose solutions they cannot find in a simple and quick way, so monitoring 24x7 tools, outside the classroom, would be desirable.

It is unanimous that the most effective method for learning any domain is practice [11]. In computer programming learning, the practice comes down to solving programming exercises. In order to have exercises it is necessary to create them or reuse existing ones, which is complicated as the best exercises are often inaccessible or in proprietary formats. Even with these exercises, it is necessary to make them available to students in an attractive and practical way, organized into well-defined thematic modules and ordered by difficulty levels. This organization and sequencing benefits the student's progress and consequent motivation [16].

However, it is not enough to put a battery of exercises for the student to solve in order to master programming. For practice to be efficient, feedback is required. If the feedback is null or inaccurate, then practice can be detrimental to the student's sustained progress. To have feedback, the teacher must have time to be able to answer all requests from the class, typically with a large number of students

Another major problem in teaching programming is that it is not personalized. Typically, teachers' strategies do not usually address all student learning styles. It is a fact that we all learn in different ways and consequently have different preferences in learning in order to assimilate content and good practices. However, by adopting traditional methods, the teacher is forcing all students to have uniform learning, at the same pace and according to their pedagogical strategies. However, the high number of students in the classroom combined with time constraints makes more personalized approaches impossible. In an optimal world, the teacher should be able to contemplate the enormous diversity of learning styles present in the classroom and adapt teaching to each of the profiles found. One solution to this problem is the use of automated tools that support this personalized teaching [10].

2.3 Psychological aspects

The cognitive and motivational aspects are fundamental to the success of learning computer programming. The lack of motivation is perhaps one of the biggest reasons for school's failure. Many students are not motivated enough to study programming, due to its reputation for being difficult and the extremely negative connotation associated with it. There are same studies that indicate that there is a public image of a "programmer" as a "social inadequate" [5,6].

In addition, students have introductory programming disciplines during one of the most difficult periods of their student life, that is, at the beginning of a college degree in computer science, coinciding with a period of transition and instability in their life. There are even authors who consider that the programming disciplines are poorly located in the curriculum [5,6].

Gamification strategies can be used in the educational process of programming learning. These strategies foster engagement through collaboration (e.g. students interact each other in order to solve a challenge) and competition (e.g. students compete to be the first to solve a challenge). In fact, new methodologies and techniques are appearing aiming to

improve retention and foster the motivation and competitiveness of computer programming learning [12]. While the concept of "winners and losers" can hinder the motivation of students [15], competitive learning is becoming a trendy learning paradigm that relies on the competitiveness of students to increase their programming skills [3] with promising results.

2.4 Difficulties in the use/integration of automated tools

Another major problem with this process is the fact that classes are typically very long which severely undermines the work of teachers. In addition to manually correct students' resolutions and give feedback, teachers have to give classes more quickly in order to teach all the course subjects and to foster the delivery of assignments to students.

These issues can be mitigated with the use of specialized online tools to support and guide the entire teaching-learning process of computer programming. Currently, there is a vast set of tools ranging from repositories of programming exercises to dynamic code evaluators [13]. However, despite the existence of several tools, their continued use is still scarce. There are several reasons for that ranging from the lack of time for its adoption and the interoperability issues in the most diverse infrastructures scenarios.

In this realm, we can organize existing tools into the following categories:

- Teaching-learning environments - environments that allow the teacher to create and manage their exercises and make them available to students and students to solve and submit them and access the resolution feedback;
- Exercise repositories - systems that allow the storage, cataloging and subsequent discovery of exercises by teaching-learning environments;
- Assessment tools - tools or services that receive the resolution for a given programming exercise and that return an evaluation of it;
- Gamification services - services that provide gamification components to be included in the teaching-learning environment with the specific aim of fostering student's engagement.

It is also important to state that computer programming learning tools are not limited to this list. Other systems can be used to assist in the process such as anti-plagiarism tools, recommendation systems, feedback animators, bots, and others.

3 Survey on Computer Programming Teaching

In order to understand how Higher Education Institutions (HEI) approach the teaching and learning processes of computer programming, particularly in the introductory units, a questionnaire survey was conducted. This questionnaire aimed to characterize what is taught and how it is taught, namely the topics covered in the unit courses, the methodologies adopted, tools, languages, good practices and the main difficulties encountered in the programming teaching process. The questionnaire ends with a request for suggestions on what might improve the teaching and learning process of computer programming. Considering these objectives, the questionnaire was organized into four sections:

- Respondent characterization (institution, course degree and course unit, number of contact hours and type of classes);
- Programming teaching (covered topics, languages and learning and pedagogical resources);
- Editors and teaching support tools (code evaluators, testing tools, plagiarism detection or gamification);
- Final considerations (average pass rate, main difficulties identified, good practices and tools that it intends to incorporate in the teaching process).

A pilot test of the questionnaire was carried out with four users. These users were invited to answer a test version of the questionnaire, providing us with suggestions for improvement. Several suggestions were received, which were incorporated in its final version.

The questionnaire was addressed to university and polytechnic higher education teachers, who teach introductory programming subjects. This target audience includes professors from higher professional technical courses (TeSP), bachelor's and master's degrees. Considering the Bologna process, some universities have created courses that combine a bachelor's with a master's, called integrated master's (five years), which are also included in the target audience.

The questionnaire was distributed through an online survey, being disseminated by eighty contacts, which fit in the target audience described above, requesting collaboration in the response. Some contacts were obtained from research carried out on the institutions' web pages, when available. Others were obtained through personal networks. The respondents were informed that the questionnaire was anonymous, ensuring the confidentiality of responses. After forty-eight hours a reminder was sent to all contacts, reinforcing the invitation to participate in this survey, which was available to respond for 10 days.

A total of 59 responses were gathered, which represents a response rate of 74%. The responses were from teachers of 4 Polytechnic Institutes (Bragança, Cávado e Ave, Oporto and Viseu) and 9 Universities (Azores, Algarve, Aveiro, Beira Interior, Coimbra, Évora, Minho, Lisbon, and Oporto). One can see that the responses obtained come from a wide-ranging geographical scope, with responses from north to south of the country, including islands (Azores).

4 Results analysis

4.1 Respondent characterization

The questionnaire was answered mostly from professors of curricular units belonging to bachelor degrees, either from Polytechnics or Universities (Figure 1). Programming classes can have four different typologies: theoretical (40.7%), theoretical-practical (57.6%), practical (18.6%) and practical-laboratory class (52.5%). More than 50% of the inquired provide 4 weekly contact hours and 20 to 25 students per class.

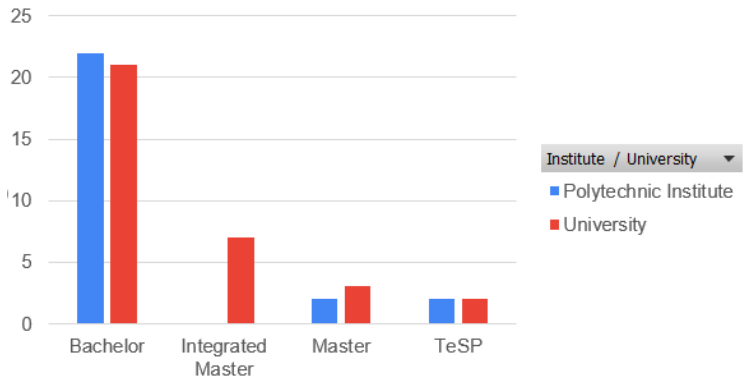


Figure 1 Degree.

4.2 Programming teaching

Regarding programming teaching (covered topics, programming languages and learning resources), the questionnaire survey offered options in a multiple-choice format. As expected, overall introductory programming curricular units covered the basics of programming, like variables (89.8%), operators (83.1%), structures and data types (89.8%), control structures (89.8%) and functions (91.5%). As for the languages used in initial programming classes, 16 of the inquired use C, followed by Python and Java (12 and 11 answers, respectively). Only 6 teachers answered C#, and the remaining languages had under 5 responses, Figure 2.

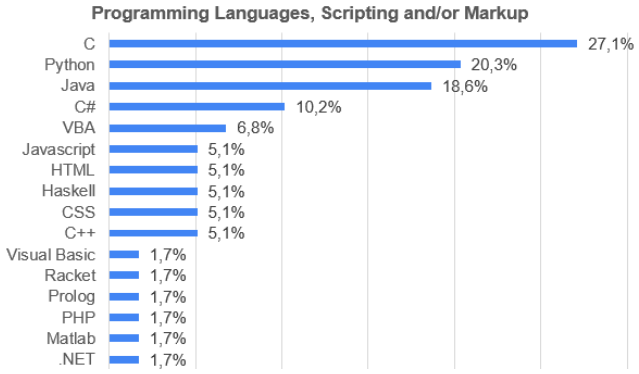


Figure 2 Programming languages.

Regarding the learning resources, mostly use classic non-interactive approaches such as presentation slides (88.1%), notebooks (25.4%) and books (72.4%). Online tools already have relevance as a resource in programming teaching, as 22.4% use online tutorials, 6.9% adopt learning platforms in their classes, like Udemy or code.org, and 5.2% make use of YouTube or other online videos. Exercise solving is a feature on which the process of teaching computer programming learning depends on. More than 85% of the professors state that the exercises are created from scratch to the curricular unit and more than 60% claim their exercises are revised each year. Almost 80% of the exercises are solved in a code editor; the remaining use some sort of platform (online or adopted to the programming curricular unit).

4.3 Editors and teaching tools

The questionnaire results revealed that a multiplicity of code editors is used in Portuguese higher education programming classes, Figure 3.

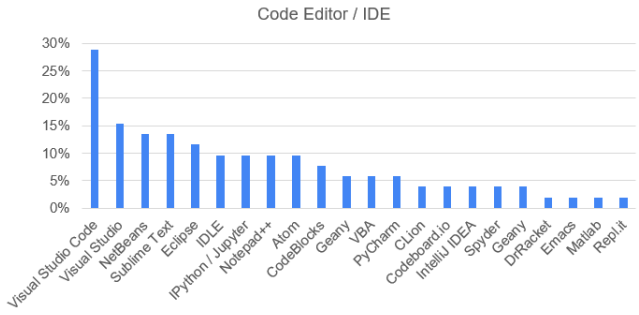


Figure 3 Code editors.

Regarding code evaluation tools, the responses obtained demonstrate that most respondents do not use code evaluators. Among those who reported using it, Mooshak stands out as the preferred option, Figure 4.

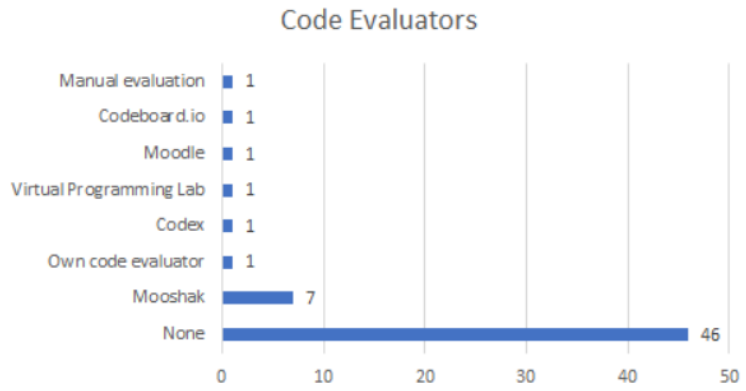


Figure 4 Assessment tools.

The same is shown when asked about testing, plagiarism detection or gamification tools. Only 4 respondents make use of some sort test framework, like Jasmine, Mocha, Enzyme, Jest, PandionJ, JUnit or QuickCheck, but none stands out as the most used. As for gamification, 6 professors employ gamification in the computer programming learning process: 2 use some tool integrated with the Learning Management System (LMS), 2 developed their own gamification and 2 others take advantage of web-based platforms like code.org and Kahoot. Plagiarism detection proved to be a major concern when compared with the latter two topics: testing and gamification. 11 answered positively when asked if they used an anti-plagiarism tool. From those, MOSS stands out as the most used (5 responses), followed, ex aequo, by Codequiry, JPlag, Urkund, Virtual Programming Lab, Blackboard SafeAssign and a proprietary application developed by the teaching staff, all of the above with 1 response each.

4.4 Final considerations

The fourth and last section of the questionnaire was composed of three open questions regarding main difficulties identified in teaching programming, best practices, and tools that could improve the computer programming teaching/learning process. More than 20% pointed out that the students' lack of strong know-how foundations and the complexity of the programming domain as the most prominent difficulties. Around 18.6% make reference to the classes with a high number of students and few contact hours (Figure 5).

Even though the average approval rate is considerably high (88% responded that the approval rate is higher than 50%), there is still potential to improve (Figure 6).

Several ideas were pointed out regarding what could improve the computer programming teaching-learning process, namely:

- More student work and responsibility
- Increase contact hours
- Motivate students (quizzes, inverted classrooms, other tools)
- Emphasis on algorithmic logic
- Introduce an automatic code evaluator
- Code execution and testing tools that provide intuitive feedback
- Smaller classes

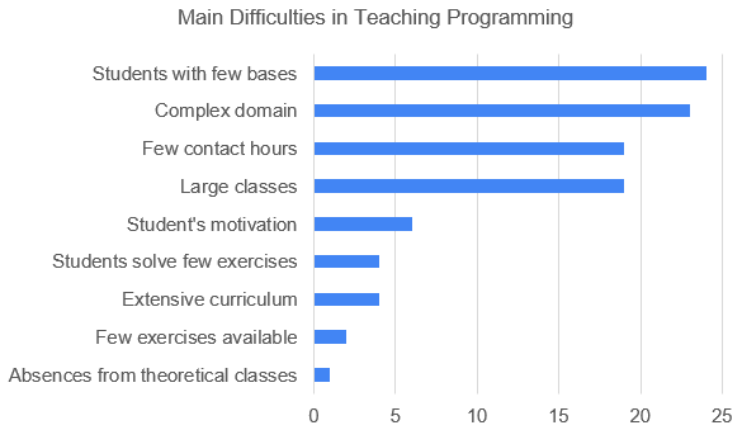


Figure 5 Main difficulties in teaching programming.

- Continuous training of teachers
- Introduce problem-based learning methodologies
- More projects
- More time for tutorial support for each student
- Introduce mob programming tool
- A pedagogical approach that motivates students
- Introductory classes with a view to standardizing students' knowledge
- Peer learning and active learning
- Guide learning to topics of interest to students
- Individual tutorial guidance
- Gamification mechanisms
- Use programming together with other curricular units

As for tools professors would like to introduce in their classes, 20 respondents did not answer or said they do not know which tool to point out. However, 29 promptly acknowledge that an automated code evaluator would improve their classes. 19 would like to include a code analyzer and 18 pointed out an open exercise repository would be a plus. Gamification, anti-plagiarism and recommendation systems were also chosen by 14, 11 and 7 of the inquired professors.

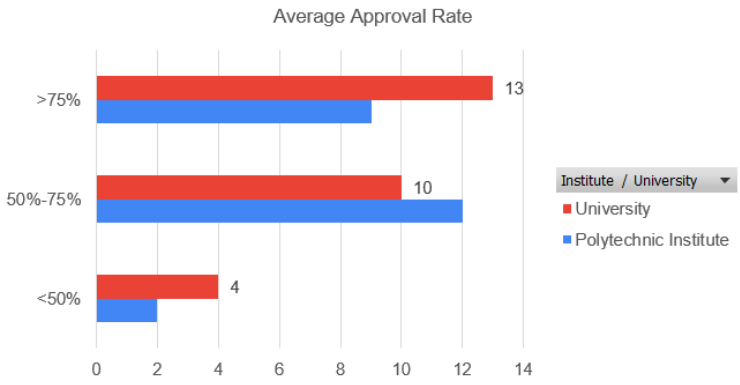


Figure 6 Average approval rate.

5 Conclusions

Learning to program is difficult. In this paper, we identify several factors that make students feel unmotivated from the methodologies used in the classroom to the psychological aspects inherent to the programming domain.

In order to try to understand how programming is taught in Portugal, a survey was carried out on more than fifty existing courses in Portugal covering a large part of the national territory and islands. The objective was to assess how programming classes are constituted, which methodologies, languages, and tools are used and what are the respondents' opinions regarding the main difficulties and which are the best approaches to solve them.

Regarding the characterization of the respondents and their classes, we had responses essentially from undergraduate courses where teachers give theoretical and practical classes for 4 hours per week.

The topics covered are initially linked to the basic concepts of languages (variables, operators, structures and data types). In terms of languages, most responses indicated C, Phyton, and Java as the programming languages taught in introductory courses. The teaching approaches are combined between slides exposing the theoretical part and the resolution of exercises in a code editor (preferably Visual Studio Code). Most exercises are created from scratch, with slight adaptations at the beginning of each year. The evaluation of the exercises is mostly done manually. In fact, the same methodology is used for testing, gamification and plagiarism detection.

Regarding the main obstacles to teaching programming, most teachers complain about the few student bases, the fact that programming is a complex field that combined with large classes and few hours of contact makes the process's time consuming and complex. Despite this, there has been a reasonable number of approvals.

As ideas for approaches to address programming learning failure, teachers point to the use of tools that automate various stages of the life cycle of the teaching process and the decrease in the number of students per class so that teaching can be more personalized.

As future work, the authors wish to improve the survey with new questions and extend the sample to international universities.

References

1 Kirsti M Ala-Mutka. A survey of automated assessment approaches for programming assignments. *Computer Science Education*, 15(2):83–102, 2005. doi:10.1080/08993400500150747.

2 Yorah Bosse and Marco Aurélio Gerosa. Why is programming so difficult to learn? patterns of difficulties related to programming learning mid-stage. *SIGSOFT Softw. Eng. Notes*, 41(6):1–6, January 2017. doi:10.1145/3011286.3011301.

3 Juan C. Burguillo. Using game theory and competition-based learning to stimulate student motivation and performance. *Comput. Educ.*, 55(2):566–575, September 2010. doi:10.1016/j.compedu.2010.02.018.

4 Micaela Esteves, Benjamim Fonseca, Leonel Morgado, and Paulo Martins. Improving teaching and learning of computer programming through the use of the second life virtual world. *British Journal of Educational Technology*, 42(4):624–637, 2011. doi:10.1111/j.1467-8535.2010.01056.x.

5 Anabela Gomes, Cristiana Areias, Joana Henriques, and António José Nunes Mendes. Aprendizagem de programação de computadores: dificuldades e ferramentas de suporte. *Revista Portuguesa de Pedagogia*, 42:161–179, 2008.

6 Tony Jenkins. On the difficulty of learning to program. In *3rd Annual LTSN-ICS Conference*, pages 53–58, 2002.

7 Derviş Kayımbaşıoğlu, Bora Oktekin, and Hüseyin Hacı. Integration of gamification technology in education. *Procedia Computer Science*, 102:668–676, 2016. 12th International Conference on Application of Fuzzy Systems and Soft Computing, ICAFS 2016, 29-30 August 2016, Vienna, Austria. doi:10.1016/j.procs.2016.09.460.

8 Jackie O'Kelly and J. Paul Gibson. Robocode – problem-based learning: A non-prescriptive approach to teaching programming. *SIGCSE Bull.*, 38(3):217–221, June 2006. doi:10.1145/1140123.1140182.

9 Ana Pacheco, Anabela Gomes, Joana Henriques, Ana Maria de Almeida, and António José Mendes. Mathematics and programming: Some studies. In *Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*, CompSysTech '08, New York, NY, USA, 2008. Association for Computing Machinery. doi:10.1145/1500879.1500963.

10 José Carlos Paiva, José Paulo Leal, and Ricardo Queirós. Authoring game-based programming challenges to improve students' motivation. In Michael E. Auer and Thrasyvoulos Tsiatsos, editors, *The Challenges of the Digital Transformation in Education*, page 602–613, Cham, 2019. Springer International Publishing, Springer International Publishing.

11 Ricardo Queirós. A framework for practice-based learning applied to computer programming. Master's thesis, FCUP, Porto, 2012.

12 Ricardo Queirós. *Gamification-Based E-Learning Strategies for Computer Programming Education*. IGI GLOBAL, 2016. doi:10.4018/978-1-5225-1034-5.

13 Ricardo Queirós. *A Survey on Computer Programming Learning Environments*, volume 1 of *1*, chapter 4, pages 90–105. IGI GLOBAL, 2019. doi:10.4018/978-1-5225-7455-2.ch004.

14 Anthony Robins, Janet Rountree, and Nathan Rountree. Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2):137–172, 2003. doi:10.1076/csed.13.2.137.14200.

15 Maarten Vansteenkiste and E. L. Deci. Competitively contingent rewards and intrinsic motivation: Can losers remain motivated? *Motivation and Emotion*, 27(4):273–299, 2003.

16 Jacqueline L. Whalley, Raymond Lister, Errol Thompson, Tony Clear, Phil Robbins, P. K. Ajith Kumar, and Christine Prasad. An australasian study of reading and comprehension skills in novice programmers, using the bloom and solo taxonomies. In *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52*, ACE '06, page 243–252, AUS, 2006. Australian Computer Society, Inc.

Learning Binary Search Trees Through Serious Games

Alberto Rojas-Salazar
Trinity College Dublin, Ireland
rojassaa@tcd.ie

Paula Ramírez-Alfaro
University of Costa Rica, Alajuela, Costa Rica
paula.ramirez@ucr.ac.cr

Mads Haahr
Trinity College Dublin, Ireland
mads.haahr@tcd.ie

Abstract

Data structures and algorithms are core topics in Computer Science, but they are difficult topics to grasp. Data structures and algorithmic concepts are abstract and difficult to relate to previous knowledge. To facilitate the learning process of these topics, learning tools that link new information with previous knowledge in an active way may be a useful approach to teach data structures and their algorithms. Furthermore, serious games have the potential to serve as a learning tool that accomplishes both objectives: to link new information with previous knowledge and to facilitate active learning. To tackle these issues, we developed *DS-Hacker*, an action-adventure serious game that utilizes the game elements to represent the Binary Search Tree (BST) properties and structure. In this paper, we report the results of a pilot experiment that compares the learning gains after completing two learning activities: (1) playing a serious game for learning Binary Search Trees, and (2) reading a summary and watching two video tutorials. Additionally, we report the results from a qualitative survey that evaluated the game usability, player satisfaction and the participants' perception about the means used by the game to deliver the BST concepts.

2012 ACM Subject Classification Applied computing → Education

Keywords and phrases Binary Search Tree, Data Structures, Serious Games

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.22

1 Introduction

Data structures and algorithms are core topics in Computer Science, and they are essential for the development of efficient software [10]. Due to the relevance of these topics, data structures and algorithms are included in the guidelines for undergraduate degree programs developed by the Association for Computing Machinery (ACM) [7]. Typically, universities teach the first introductory data structure course in the second year of their undergraduate Computer Science programs [8].

While a deep understanding of data structures is fundamental knowledge for computer scientists, advanced data structures and their algorithms are difficult topics to grasp [2]. Data structures and algorithmic concepts are abstract and difficult to relate to previous knowledge. From a constructivist point of view, it is important that new experiences and information link to previous knowledge in order to create new knowledge [6]. Educators should provide experiences and environments where the students can construct knowledge through reflection, critical thinking and their previous knowledge [6]. Therefore, learning tools that complement classes by linking new information with previous knowledge in an active way may be a useful approach to teaching data structures and their algorithms.

First International Computer Programming Education Conference (ICPEC 2020).
Editors: Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões; Article No. 22; pp. 22:1–22:7

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In view of the above, serious games have the potential to serve as a learning tool that accomplishes both objectives: to link new information with previous knowledge and to facilitate active learning. Many game genres are popular among teenagers and young adults, and well-crafted video games promote active learning [5]. Usually, video games offer challenges that require active engagement of the player. Therefore, serious games may take advantage of these characteristics in order to facilitate the association of new information with previous knowledge and active learning.

In this paper, we present a serious game for teaching Binary Search Trees (BSTs) called *DS-Hacker* (Data Structure Hacker). *DS-Hacker* aims to introduce BST concepts to college students by means of relating well-known game elements with BST concepts. These relations are presented to the learner through analogies embedded in the game. We also report the results of a pilot experiment that compares the learning gains after completing two learning activities: (1) playing *DS-Hacker*, and (2) reading a summary and watching two video tutorials. Finally, we report the results from a qualitative survey that evaluated the game usability, player satisfaction and the participants' perception about the means used by the game to deliver the BST concepts.

Results show that both learning approaches produces a learning effect and that there is no statistically significant difference between both activities. The qualitative survey suggests that participants perceived that they learned while playing the game, and that they could relate the BST concepts and structure with the *DS-Hacker* game elements.

2 DS-Hacker

DS-Hacker is a PC game developed with Unity 3D, and its target population are university students from Computer Science and Engineering Schools. The game is a third person 3D action-adventure game, a well-known genre. The aesthetics are sci-fi style, and its story takes place in a distant future where a corrupt corporation is harming the balance of society. In the game, the player takes the role of the robotic hacker that must traverse a maze composed of chambers and extract the information stored in the maze. A video file of the gameplay of the English version is available through the following link: `https://www.dropbox.com/s/8vavy0e7b9uywx6/DS-Hacker_Level1%26Level2.mp4?dl=1`

To achieve learning of the BST structure, *DS-Hacker* uses an analogy between the BST structure and the environment structure. According to the game plot, corporations hide and protect their information in places called "Data Systems" (our game environment). Data systems are mazes organized as well-known data structures, and to achieve our teaching objective, the Data System reflects the structure of a BST. Therefore, many elements of the game environment represent the most important elements of the data structure. For instance, in *DS-Hacker*, the maze's rooms represent the nodes; the portals of each room represent the links that points to other nodes; the room ID represents the comparable key; and the information stored in each room represents the associated values of each node. Furthermore, the chambers of the maze are organized following the BST property.

The game story serves a major function because it delivers the conceptual knowledge. The game story is delivered through a non-player character (NPC) named Anonymous who always appears at the beginning and at the end of each level. Anonymous introduces the missions and the necessary BST concept to accomplish them. In order to facilitate the understanding of the BST concepts, Anonymous takes advantage of analogies between the game elements and the BST elements. For instances, in the first two levels, Anonymous informs the player about the relation between the game environment structure and the BST structure. In the last three levels, Anonymous presents the relation between the game challenges and the search algorithm.

Currently, *DS-Hacker* possesses five levels, and each level focuses on different concepts of the BST data structure. Level one and two cover topics related to the basic structure of the BST, its properties, and the structure of its nodes. Level three, four and five cover topics related to the search algorithm such as the sequence of the algorithm's steps and its outcomes. Furthermore, each level has a mission, and each mission possesses one or more challenges. Missions provide opportunities to apply and solve problems using the concepts given by Anonymous and to experience the structure of the BST in a concrete manner.

3 Method

Our pilot experiment follows a "switching replications" experimental design. In the study, participants are randomly divided into two groups, G1 and G2. Both groups must complete two activities (the treatment and the control activity) and answer three tests (pre-test, mid-test and post-test). The study is organized as follows: First, all participants take a pre-test; then G1 performs the treatment, and G2 performs the control activity; then, all participants answer the mid-test; then, G1 and G2 switch and perform the other activity; finally, all participants take the post-test. Switching replications design may decrease social threats to validity, since it allows all participants equal access to the treatment activity [4]. However, the learning effect due to the overexposure to the test may lead to a testing threat [4].

The experiment was carried out as a workshop during a class of the 2020 Summer Term in University of Costa Rica. Participants were randomly divided into two groups and assigned to a computer with the game already installed. Participants of G1 played the Spanish version of *DS-Hacker* (the treatment); meanwhile, participants of G2 completed the control activity. Then, G1 performed the control activity, and G2 played the game.

The control activity included two popular teaching methods: a written summary of the BST concepts and three video tutorials. The summary was a Spanish translation of the book *Algorithms* by Sedgewick and Wayne [10]. The first video tutorial[1] was about BST structure and characteristics, and the second[2] was about the search and insert operations (the insert operation was not evaluated). The third video[3] tutorial was a general summary about the BST basic concepts and operations.

The pre-test, mid-test and post-tests were designed to assess the learning gains. The tests have 23 questions and cover the first four levels (remember, understand, apply, and analyse) of the revised version of the Bloom's taxonomy [1]. The questions were multiple-choice, and their construction followed the guidelines suggested in [9]. Furthermore, the questions verify factual, conceptual and procedural knowledge. Besides the tests, participants took a demographic survey at the beginning of the experiment and a qualitative survey to evaluate the game at the end of the experiment. All surveys (and tests) were performed using Google Forms.

Initially, 32 students participated in the experiment; however, we excluded 5 participants from the analysis because they did not complete one of the tests. Therefore, we only take into consideration the 27 participants who completed all the evaluations. Group 1 (the experimental) had 13 participants, and Group 2 (the control) had 14. In terms of background, 11 students were from the Computer Science School; 10 students were from the Industrial Engineering School; and 5 students from the Electrical Engineering School and the Mathematics School.

[1] https://youtu.be/Bh61AvHAf90
[2] https://youtu.be/DVKDQcJOqy8
[3] https://youtu.be/mTMrszfrNtI

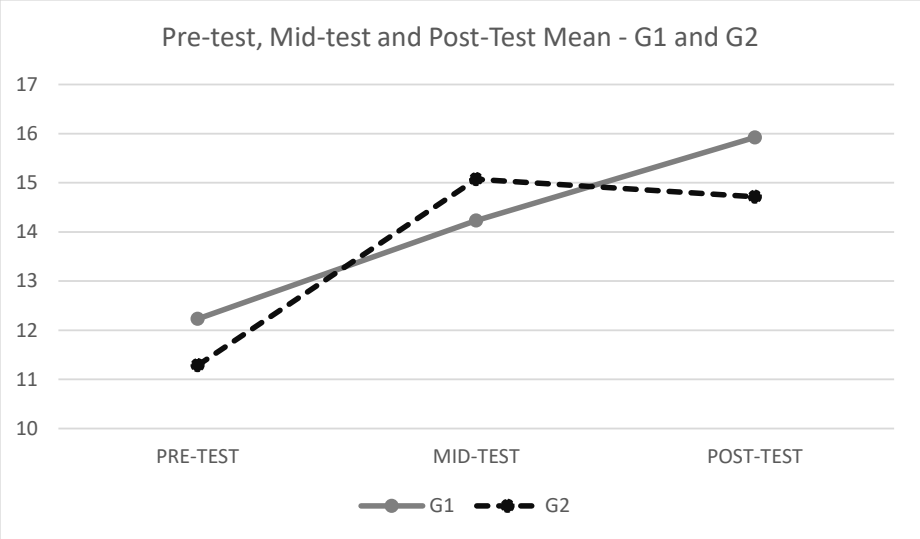


Figure 1 Mean of the pre-test, mid-test and post-test of G1 and G2.

Table 1 G1 and G2 Pre-test, Mid-test and Post-test means and standard deviation.

	G1 Pre-test	G1 Mid-test	G1 Post-test	G2 Pre-test	G2 Mid-test	G2 Post-test
Mean	12.231	14.231	15.923	11.286	15.071	14.714
StDev	3.395	2.948	2.691	3.832	3.452	3.730

4 Results and Discussion

Figure 1 shows the average of the scores obtained during the pre-test, mid-test and post-test of G1 and G2. The chart presents a learning effect for both activities. After the first round of activities, G2's participants performed better in the mid-test than G1's participants. On average, G1's participants (who played the game) increased by 2 points. Meanwhile, G2's participants (who watched the video tutorials) increased by 3.79 points. After switching and completing the second round of activities, G1's participants performed better in the post-test than G2's participants. G1's participants increased by 1.69 points; G2's participants decreased 0.36 points. After both activities, G1's participants increased by 3.69 points, and G2's participants increased by 3.43 points. Table 1 presents the mean and the standard deviation of each test.

Additionally, we performed a t-test analysis to verify whether the difference between the means of the pre-test, mid-test, and post-test were statistically significant. Table 2 presents the results, showing no significant difference. We also verified whether the difference between mid-test and pre-test and post-test and pre-test of G1 and G2 were statistically significant. To achieve this, we per-formed a series of two-tailed paired t-tests with an alpha of 0.05. In addition, to verify the magnitude of the difference, we calculated the Cohen's d that quantifies the effect size. In our case, it determines the magnitude of change in scores. Cohen's d result larger than 0.80 is considered a large size effect; a result around 0.50 is considered a medium size effect, and around 0.20 a small size effect [3]. Table 3 shows the results of the calculations.

Table 2 Mean, p-value and t statistic of the difference of scores.

	N	Pre-test	Mid-test	Post-test
Group 1 - Mean	13	12.231	14.231	15.923
Group 2 - Mean	14	11.286	15.071	14.714
Difference P-Value		0.503	0.502	0.341
T Statistic		0.679	-0.682	0.971

Table 3 T Statistic, p-value, and effect size of the difference between Mid-test and Pre-test, and Post-test and Pre-test.

	P Value	T Statistic	Effect Size
Difference: Pre-Test and Mid-test of Group 1	0.0218	-2.6331	0.63
Difference: Pre-Test and Mid-test of Group 2	0.000009	-5.7303	1.04
Difference: Pre-Test and Post-test of Group 1	0.000095	0.7012	1.21
Difference: Pre-Test and Post-test of Group 2	0.000094	-5.5511	0.91

The previous results indicate that both learning activities were effective. The differences between pre-test and mid-test of G1 and G2 are statistically significant. However, the results show that the control activity (reading and watching video tutorials) was more efficient than playing *DS-Hacker*. For instance, the effect size of the difference between the mid-test and pre-test of G2 is higher than the size effect of G1. Additionally, the average score of the mid-test of G2 is almost the same as the average score of the post-test of G1. Even though, the difference between mid-test and post-test averages are not statistically significant.

Another interesting finding is that G2 slightly decreased its performance during the post-test (after playing the game) and that G1 increased the scores in the mid-test and post-tests. This discovery suggests that the order of the treatment may affect the learning gains. Further studies regarding the order of the learning activities may lead to promising results.

The qualitative survey has 19 four-point Likert-scale questions divide into three categories. The first category (Q1-Q6) assesses the participant's perception about learning and the means used by the game (environment, story and challenges) to deliver the BST concepts. The second category (Q7-Q15) assesses the usability. The third category (Q16-Q19) assesses the enjoyability. We only present the answers of the 27 participants who completed all the tests. Table 4 presents the percentage of the positive answers ("Strongly agree" and "Moderately agree") of the qualitative survey.

Most of the answers of the qualitative survey were positive. However, three questions received a considerable number of negative responses, indicating that the game has some problems. Q3 responses indicate that half of the participants could not understand the search algorithm principles while playing the game. This is an indication that we should improve the content and levels that cover this topic. Second, Q8 and Q9 answers suggest that participants had trouble dealing with the game controls. The cause of this problem was the low performance of the computers utilized during the experiment. We should consider this factor, and we should optimize the game to make it appropriate for low performance computers.

Regarding the learning approach of the game, participants reported that they could relate the BST concepts with the game environment and the story. Additionally, results indicate that participants think that the game provides an environment that allows them to practice the BST concepts. Finally, participants reported that they felt that they were learning while playing the game, and that in general, they enjoyed the game.

Table 4 Distribution of the positive answers of the qualitative survey.

Questions	Agree %	Questions	Agree %
Q1. The game help me to understand BST structures.	77.78	Q11. The game tutorial was useful and clear.	81.48
Q2. The game help me to understand the nodes' structure.	81.48	Q12. The voice and way of talking of the NPC ware clear.	81.48
Q3. The game help me to understand the search algorithm.	59.26	Q13. Game missions were clear.	81.48
Q4. I could relate concepts presented in the game story with the BST concepts.	88.89	Q14. The game GUI was easy to un-derstand and intuitive.	85.19
Q5. I could relate the game environment with the BST structure.	85.19	Q15. The game menu has useful op-tions.	81.48
Q6. The game allows me to practice the previously learned BST concepts.	85.19	Q16. I enjoyed playing DS-Hacker	70.37
Q7. The game was easy to learn.	85.19	Q17. I like the way that BST concepts were presented during the game.	81.48
Q8. The game controls were easy to learn.	66.67	Q18. I think that video games increase my motivation towards computer science topics.	85.19
Q9. The game controls respond smoothly.	48.15	Q19. I would like more serious games to be used to teach data structures.	81.48
Q10. The map was easy to understand.	77.78		

5 Conclusion and Future Work

The article also presented the results of a pilot experiment and a qualitative evaluation of *DS-Hacker* which aims to facilitate learning of the BST data structure and associated algorithms by linking new information with previous knowledge and facilitating active learning. The results of the pilot experiment show that the treatment (playing the game) and the control activity (reading a summary and watching video tutorials) produced learning gains on the participants. Differences between the scores obtained by the treatment group and the control group were not statistically significant. However, results from the mid-test suggest that the control activity is slightly more efficient than playing the game.

Our qualitative evaluation showed that the participants could relate game elements (game story, environment and challenges) with the BST concepts. This finding suggests that these game elements may be used to delivery educational information. Additionally, participants felt that they learned while playing the game. Regarding the usability of the game, we must optimize the game to run on low performance computers. In general, participants reported that they enjoyed playing the game.

In the future, we plan to redesign the levels that cover the search algorithm and add more level covering other BST operations such as the tree traversal algorithms and the insert algorithm.

References

1. Lorin W. Anderson and David R. Krathwohl. *A taxonomy for learning, teaching, and assessing: a revision of Bloom's taxonomy of educational objectives*. Longman, New York, 2001.
2. Katrin Becker and Melissa Beacham. A Tool for Teaching Advanced Data Structures to Computer Science Students: An Overview of the BDP System. In *Proceedings of the Second Annual CCSC on Computing in Small Colleges Northwestern Conference*, pages 65–71, USA, 2000. Consortium for Computing Sciences in Colleges. event-place: Oregon Graduate Institute, Beaverton, Oregon, USA. URL: http://dl.acm.org/citation.cfm?id=369274.369319.
3. D. Dicheva and A. Hodge. Active Learning Through Game Play in a Data Structures Course. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, pages 834–839, New York, NY, USA, 2018. ACM. event-place: Baltimore, Maryland, USA. doi:10.1145/3159450.3159605.
4. Michael Eagle and Tiffany Barnes. Experimental Evaluation of an Educational Game for Improved Learning in Introductory Computing. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, SIGCSE '09, pages 321–325, New York, NY, USA, 2009. ACM. event-place: Chattanooga, TN, USA. doi:10.1145/1508865.1508980.
5. James Paul Gee. *What Video Games Have to Teach Us about Learning and Literacy*. Palgrave Macmillan, New York, 2nd edition, 2007.
6. Aytac Gogus. Constructivist Learning. In Norbert M. Seel, editor, *Encyclopedia of the Sciences of Learning*, pages 783–786. Springer US, Boston, MA, 2012. doi:10.1007/978-1-4419-1428-6_4049.
7. Association for Computing Machinery (ACM) Joint Task Force on Computing Curricula and IEEE Computer Society. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. ACM, New York, NY, USA, 2013.
8. R. Lawrence. Teaching data structures using competitive games. *IEEE Transactions on Education*, 47(4):459–466, November 2004. doi:10.1109/TE.2004.825053.
9. Rafael Moreno, Rafael Martínez, and José Muñiz. Directrices para la construcción de ítems de elección múltiple. *Psicothema*, 16(3):490–497, 2004.
10. Robert Sedgewick and Kevin Wayne. *Algorithms*. Addison-Wesley, 4th edition, 2014.

IoEduc – Bring Your Own Device to the Classroom

Miguel Silva
IOTech – Innovation on Technology, Porto, Portugal
University of Porto, Portugal
miguelsilva@iotech.pt

Diogo Ferreira
IOTech – Innovation on Technology, Porto, Portugal
University of Minho, Braga, Portugal
diogoferreira@iotech.pt

Filipe Portela
Algoritmi Research Centre, University of Minho, Braga, Portugal
IOTech – Innovation on Technology, Porto, Portugal
cfp@dsi.uminho.pt

Abstract

The evolution of technology brings new challenges to teaching platforms. Students are demanding and want to have dynamic and interactive environments. Classrooms are much more than a simple place to study. Students can work and learn anywhere and anytime they want. Having this reality in consideration emerges the concept of Bring Your Own Device to classrooms. So, in the first instance, it is essential to understand the concept and which tasks can be transported to students devices. After that, it is time to design a new tool capable of answering new teaching demands. ioEduc arises as a platform able to support a unique and interactive way of teaching. This emerging platform was tested during a semester, and the assessment results of utility are promising (0% of negative answers and more than 60% of students consider this tool indispensable to the future).

2012 ACM Subject Classification Information systems → Enterprise applications

Keywords and phrases BYOD, ioEduc, Education, e-Learning, Interactive classes, Gamification

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.23

Acknowledgements I want to thank IOTECH for supporting the project.

1 Introduction

Nowadays students find the need for a different type of learning approach in order to find a motivation to expand their knowledge [2]. With the evolution of technology all the processes that used to be on paper are now becoming digital. This transition is also present in education and is impacting the way of how the students learn and study [9]. Gamification is one way to help students to find motivation to learn and work harder in classes [1].

Even though there are already solutions and platforms to encourage e-learning, these still have weaknesses, such as the difficulty of being used in several types of devices and receive information in real time, which is really important in order to follow Bring Your Own Device (BYOD) concept since students should be able to use this platforms regardless of the device they want to use [3]. With this e-learning platforms being available to everyone use in their personal devices some concerns arise, the biggest of them is data privacy, a concern that is transversal to all digital services, although is missed in many of these e-learning platforms (e.g. Teachers publish students evaluations on platforms and every student has permission to see the evaluation of his colleagues).

ioEduc arises in this context and it wants to address these concerns and create an innovative and interactive way of learning where students can access all the materials used in classes, play real time subject related quizzes, evaluate their workgroups anonymously, and view only their grades, not compromising the privacy of the remaining students.

This paper has the goal of presenting an innovative and interactive platform which personifies the concept of BYOD to the classroom. This platform was used in a real context at University of Minho and the results are motivating. This paper is divided into six sections, the first one is an introduction to the problem itself, providing some concepts to take into account during this document. The second section provides the background concepts like BYOD and gamification, having also a subsection referred to related work, in order to understand the problematic. The third section talks about the ioEduc platform, their features and own it improves the teacher work and the student's learning process. The fourth section shows a SWOT analysis of ioEduc platform, trying to understand the ioEduc capabilities and the problems that may appear. The fifth section shows the result of a questionnaire performed at University of Minho during the first semester of 2019/2020 in a subject that used ioEduc, in order to understand the receptivity to this platform. Finally, on the sixth section there is a conclusion of this document, trying to summarize the ideas shown earlier.

2 Background

2.1 Bring Your Own Device

According to Moreira et al. [11] Bring Your Own Device (BYOD) is a subset of the consumerization of Information Technologies (IT) as private or personally owned IT resources, such as computer devices or software that are used for business purposes. In education, BYOD consists of bringing laptops, smartphones or other devices to the classroom in order to increase active learning [6].

ioEduc implements the concepts of Bring Your Own Device to classrooms as it is a multi platform app that can be accessed through students personal devices.

2.2 Gamification

Gamification can be defined as "using game-based mechanics, aesthetics and game thinking to engage people, motivate action, promote learning, and solve problems" [7]. The main purpose of gamification is to merge non-gaming environments with gaming components, particularly points and rewards. The use of game mechanisms as a method of learning has been discussed since 1938. Although gamification has only started to be a widespread technology trend since 2010 [1, 4].

Education is one of the most important factors for the development of any country [1], so it is important to have students motivated and prepared to deal with pressure. Gamification plays an important role in helping students who need to be motivated as they'll have a goal to achieve they'll work harder and hit the deadlines, boosting their skills, and confidence [5].

The ioEduc platform allows the teacher to assign yellow cards to students that don't show any interest or relevant work during the semester. After a certain number of yellow cards, pre configured by the teacher, the student can get a red card. With this feature the teacher have a record of work and evolution of each student. That can help to either alert the student that his performance is insufficient or the teacher if the student evolved during a certain period.

2.3 Related Work

ioEduc is not the first solution that encourages students to bring their own devices to the classroom [8]. Moodle is an e-Learning platform made for students and teachers where they can write summaries, upload documents and helpful materials, create evaluation moments where students can submit their work and view their grades later. Another solution is Blackboard Learn, which is a virtual learning environment and learning management system. It is a Web-based solution with features of course management, customizable open architecture, and scalable design.

Although both solutions are well-known and much-used, none is fully adapted and optimized to mobile devices.

3 ioEduc

Nowadays, people tend to increasingly use their mobile devices on every task and at every moment of the day. The concept of being always connected is a reality that does not seem to change easily. That is even more evident in the younger age groups, the current students, those who were raised in a generation where there is an app for everything and all they need is a click away.

Thus, why not take part in this phenomenon and use it to create an engagement with the students? It is essential to create a new and interactive teaching environment capable of motivating students in computer science classrooms [10].

Based on this opportunity, the ioEduc platform was born. This platform was designed to be used in education, aiming to support both teachers and students. That objective is achieved by allowing the users to access all the information of their subjects since the class slides, files that the teachers wants to share, evaluations, presences register, groups creation and evaluation, and much more functionalities that will be explained later.

At the same time, ioEduc fills a gap that we can find in others e-learning platforms, the lack of privacy of each student grades, the ability to let the students mark presence in the classes using their mobile devices and the ability to contact the teacher via chat and expose any doubt that the student can have, even during the classes, allowing the students to overcome shyness and and eventually increase their knowledge.

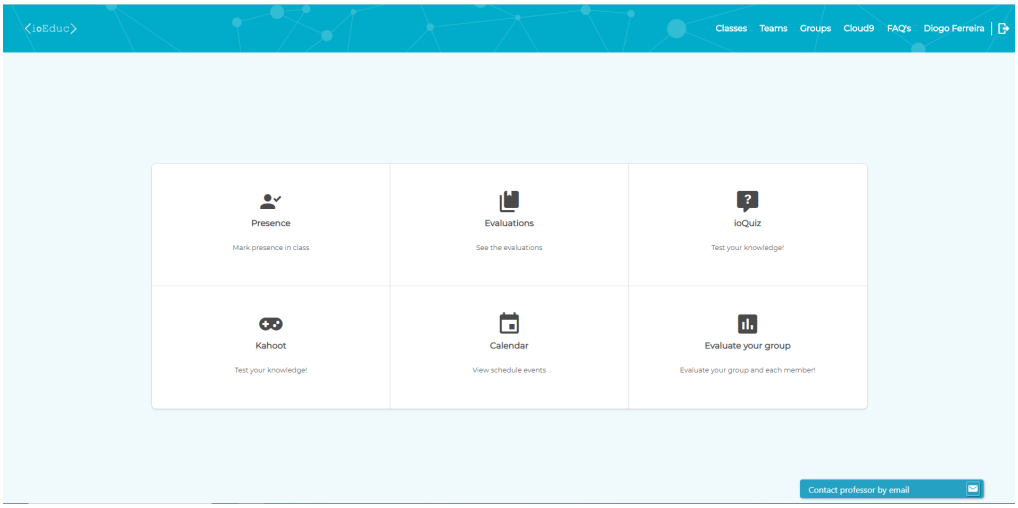


Figure 1 First page of the student's account.

Table 1 ioEduc main features and users access.

Feature	Teacher	Student
Mark presence on the class		X
See presences and assign bonus	X	
See class slides and other resource files	X	X
Manage slides and other resource files	X	
See software credentials	X	X
Manage software credentials	X	
See FAQ's	X	X
Manage FAQ's and FAQ's categories	X	
Create evaluation moments	X	
Submit student's grades for each evaluation moment	X	
See student's grades at each moment	X	
See only the logged user grades		X
Create and see quizzes on ioQuiz tool and the results	X	
Submit and see the logged user ioQuiz grades		X
Create Kahoot quizzes and see the results	X	
Submit Kahoot quizzes		X
Create projects and teams	X	
Create groups	X	X
Evaluate group and each group member		X
See the logged user's group evaluations	X	X
See all the group evaluations	X	
Assign penalization to students	X	
Create calendar events to a subject	X	
See calendar events of a subject		X
Manage Live Class	X	
Interact with live class	X	X
Ask questions on the integrated chat		X

So, we can say that ioEduc is an innovative, adaptive and user-friendly platform that can be a valuable asset in education because it provides several tools that can simplify the related parties lives. As referred earlier, ioEduc aggregates multiple modules and functionalities, in order to provide a solid working platform.

As seen on Figure 1, the first page that the students see when they are logged on ioEduc platform, the design is really simplistic, in order to make the user experience as smooth and pleasant as possible. However, although the design is really simplistic, the platform has a lot of features that can help both students and teachers on a daily basis.

Some of the main features are introduced in Table 1, showing also which type of users have access to them.

ioEduc is a tool created to answer the TechTeach metodology [10]. Thus, the side by side comparison with other LMS (Learning Management Systems) wouldn't represent any advantage to the context of this analysis.

The next subsections present some of the the capabilities of ioEduc and how they benefit their users.

3.1 Use your own device to mark presence at the classes

ioEduc allows the students to mark their presence in the classes, by inserting a unique class code in a specific timing defined by the teacher. Later, the teacher can see the students that attended the class and can also give an aleatory bonus to them. With this module, the way of validating their attendance makes the process more reliable, since one student cannot mark the presence of another student in a simple way and in the given time; is a much faster process than the traditional piece of paper to record the presence or the teacher calling each student name. Besides, with the possibility to bonify an aleatory student that attended the class, the students will feel more motivated to go to the classes.

So, this module for itself, by having the capability to increase the attendance to the classes can consequently improve the students results at the subject.

3.2 See your evaluations

As seen in other e-learning platforms, the teachers are capable of releasing the students grades. These releasings have a privacy issue, because the students can see each other grades, leading to comparisons between them, conflicts and eventually problems with the teacher's criterion.

With the ioEduc platform, the teachers can create multiple evaluation moments, that can be quizzes, exams, group evaluations. The teacher can evaluate each student individually or he can upload a CSV file with the student institutional identification and her grade. Then, each student can only see the evaluation that concerns him, mitigating the privacy problem explained earlier.

3.3 See frequently asked questions and credentials to subject's software

In order to facilitate the teacher's work, ioEduc provides a way to create and manage frequently asked questions, commonly known as FAQ's. This feature facilitates the teachers work because if he sees that the students have common doubts, rather than respond to all of then the same answer, he can create a new FAQ and if he wants to, he can organize them in categories, facilitating the student's search.

Similarly, the management of the student software credentials is more simple since they can consult the ioEduc credentials area to access them, without the need to ask the teacher and make him search between all the student credentials.

3.4 See scheduled events

This module is really simple, however it can play an important role in a subject organization, because all the official events (e.g. classes, topics by week, assessments, milestones, presentations, among others) are scheduled on the subject's calendar in order to avoid confusion with multiple events and have all of them organized and quickly available.

3.5 Make quizzes with ioQuiz and Kahoot

Another great ioEduc tool is called ioQuiz, and is a tool that allows the teacher to create quizzes in a simple and quick way, allowing the students to answer them. Since the ioQuiz tool is integrated with ioEduc, all the process is transparent and both teachers and students can see the quiz results on the evaluations page. If the teacher chooses to, the bonuses earned

with the presences (explained when we talked about the mechanism to mark presence at the classes) can be applied to the quiz results, allowing the students to have extra points and eventually improve their final grades. ioEduc is also integrated with Kahoot, a live quiz platform, allowing the students to interact more and test their knowledge during the classes.

3.6 Manage projects, teams, groups and assess you group

The group management tasks are the most time consuming to teachers because they need to register the projects, the work groups and even the assessment of each student to his own group. ioEduc provides a solid way to manage all this process without all that work. The subject's teacher creates the projects that the students can choose, then he can associate each project to a team. The group creation can be made by the students, by choosing the group leader, choosing each element role and their project. Later on, if the teacher creates evaluation moments to do so, each student evaluates their group and each element individually (in a grade between N-4 and N+4), reporting if each member worked and collaborated as expected, and giving each element a grade based on their collaboration.

To avoid problems with the evaluations and in order to have the truth about what happened during the project without any constraints, all the evaluations are anonymous, although they can be traced back by the teacher if he really needs to. The teacher can always see the group assessments, having an overview of the average evaluation of each group. If he wants to get more details, he can see the average individual assessment made by the group colleagues and the number of times that each one is marked as if not collaborated as expected. If the teacher considers it appropriate, he can penalize the student.

The penalization system works with four card colors, without any penalization the students have a green card, the teacher can penalize a student and after that, green card becomes a yellow card. If the student has another penalization the yellow card becomes an orange card, and after that, any penalization means that the student has a red card and if the teacher wants to, he can fail the student. This feature allows the teacher to maintain a full report of each student's evolution during the project.

3.7 Online chat and offline messages

The ioEduc platform also has an integrated chat, allowing the students to communicate with teachers and expose their doubts. If teachers aren't online to answer the students, an email will be sent to them with the student's question. One of the particularities of this chat solution is that the questions can be sent to the teacher during the classes, allowing the interaction and the clarification of the questions in real time. This one turns into a more important feature when applied to shy students that are not comfortable to talk in public.

3.8 Live class

The live class module is an area where the teacher can access all the resources that he needs during the classes. This module allows the teacher to select interactive slides to present to the class, videos, it has a clock and an alert and notification system. The teacher can also see the students that are present at the class and assign bonuses or even penalize a student directly from this page. This module also allows the creation and presentation of Kahoot quizzes, making the students interact more in the classes. Another interesting feature is a noise meter that is a tool that alerts both teacher and students that the amount of class noise is critical.

4 SWOT Analysis

ioEduc is an in-development platform, so it is crucial to do a SWOT (Strengths, Weaknesses, Opportunities and Threats) analysis as can be observed in Figure 2.

The major strengths of ioEduc are the possibility of students bringing their own devices to the classroom and validate their attendance, which could be really helpful in order to replace the old methods like signing and card validation. One of the biggest opportunities of ioEduc to grow is by implementing it in more universities so students and teachers can bring their own device to classes and start to have more interactive classes. ioEduc also has weaknesses and threats, the biggest weakness is that it is not possible to access the information offline, so if there is no internet some classes could be compromised.

Some threats can be faced by ioEduc, universities could start to develop and implement their own systems or collaborate with existing platforms in order to have a more robust solution that could have more feature overtime.

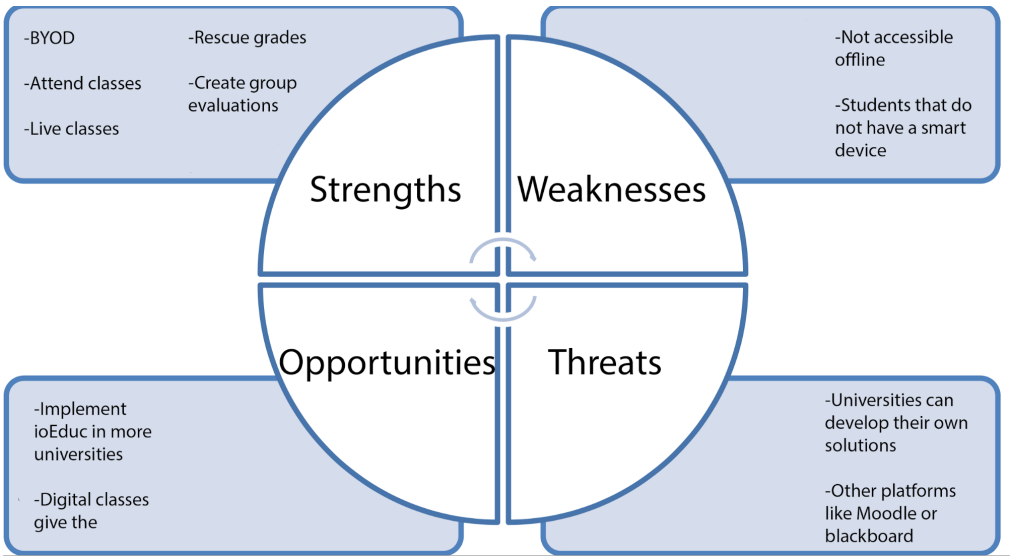


Figure 2 SWOT analysis of ioEduc platform.

5 Results

ioEduc was exploited at the University of Minho during the first semester of 2019/2020. During this period professor used this platform to turn the classes of Web Programming more interactive and attractive. More than One Hundred of students used this tool, and the feedback received is too positive. A semi-structured questionnaire was designed by the leading professor to assess and understand the impact of the platform. This questionnaire was disseminated near the students on the last lesson of the semester during two phases:

1. Structured questions:
   - Should ioEduc be a gamble?
   - Do you approve the concept of BYOD in the classroom?
2. Open Question. Give your opinion regarding the platform (positive / negative / or aspects to consider)

Seventy Three students answered to phase one and One hundred and fifty four students participated in phase two.

Table 2 Student's answers distribution.

	No	Maybe	Yes	Definitely Yes
Q1	0,00%	39,73%	30,14%	30,14%
Q1	4,35%	–	95,65%	–

The second phase was used to understand the real opinion and understand their answers in step 1. All the students answered step two. After analysing the responses, it was possible to observe that the students who put maybe in phase one are associated with some external issues like internet:

The second phase was used to understand the real opinion and understand their answers in step 1. All the students answered step two. After analysing the responses, it was possible to observe that the students who put maybe in phase one are associated with some external issues like internet:

- Slow to load pdf images
- the application often went down in the middle of the class which made the monitoring of the taught pp more complicated
- I think the only things to improve are small aspects like lags, session and login failures.
- Good, simple platform, it just takes a while with the university internet.

As the most positive aspects, it is important to highlight:

- The platform is well organized and constituted, interactive, of interest to students, good dashboard and good information regarding student data, study methods etc.
- Very well designed and encourages programming.
- Very educational and accessible.
- It is a well accomplished and very interactive platform. I loved the idea of online presence booking.

At last, it is fundamental to mention that the ioEduc was developed at the length of the semester. Several improvements and optimisations performed in "real-time" after receiving the users (students) feedback. This fact can be proved by the student analysis:

- It is still in development and therefore some bugs are seen. However, the platform itself is a good initiative.
- The platform is improving more and more. At first I found some bugs but recently it is much more stable and works most of the time.
- The idea is good, I think it was well incorporated by the teacher, despite the 'bugs' registered in the first weeks.

6 Conclusion

The evolution of technology created several opportunities in almost every working area. In the education area there are only a few platforms that try to mitigate this lack of solutions. However, they have some issues like data privacy and their resources should be improved in order to create a bigger participation and engagement of the students. With the new technology reality, the idea of having all the information on your mobile devices, concepts like BYOD (Bring your own device) and gamification could enhance the interaction with the students and eventually increase their grades.

The ioEduc platform is a software built to turn these concepts into a reality. It was built taking into account the needs of both the teachers and the students, having features that goes from using the students own device to mark presence at the classes and get a bonification or, if the teacher wants to, a penalization, turning the gamification concept into a reality,

see the evaluations individually in the case of the students, see frequently asked questions and credentials to subject's software, see scheduled events, make quizzes with ioQuiz and Kahoot, manage projects, teams, groups and assess groups and each member individually and anonymously, online chats and offline messages and a live class module to allow teachers to have all the tools needed to the classes without opening multiple pages.

This platform was exploited at the University of Minho during the first semester of 2019/2020 [10] where a semi-structured questionnaire was designed by the leading professor to assess and understand the impact of the platform. The student's receptivity to this platform was very positive, giving the idea that a vast majority of them (60%) approved this platform to use on a daily basis. Taking this into account, ioEduc has potential to become widely used and to grow, using the concepts as BYOD and gamification to improve the interactions at the education area and consequently the student results. The gamification is accomplished by the cards system, allowing teacher to reward or to punish the student according to his interest and work.

Although this platform represents a practical tool that aims to generate value to the society by helping the educational process, it can act like an inspirational agent to the development of new solutions applied to this area.

In the future, this platform will be optimised (e.g. offline mode), and some of the problems reported by the students will be solved. Then, it will be time to do some partnership to explore ioEduc in other institutions.

References

1 Patrick Buckley and Elaine Doyle. Gamification and Student Motivation. *Interactive Learning Environments*, 2014. doi:10.1080/10494820.2014.964263.

2 Lee Yen Chaw and Chun Meng Tang. What makes learning management systems effective for learning? *Journal of Educational Technology Systems*, 47(2):152–169, 2018. doi:10.1177/0047239518795828.

3 Carolina Costa, Helena Alvelos, and Leonor Teixeira. The use of moodle e-learning platform: A study in a portuguese university. *Procedia Technology*, 5:334–343, 2012. 4th Conference of ENTERprise Information Systems – aligning technology, organizations and people (CENTERIS 2012). doi:10.1016/j.protcy.2012.09.037.

4 Darina Dicheva and Christo Dichev. Gamification in Education: Where Are We in 2015? In *Gamification in Education: Where Are We in 2015?*, 2015.

5 Darina Dicheva, Christo Dichev, Gennady Agre, and Galia Angelova. Gamification in Education: A Systematic Mapping Study. Technical Report 3, Educational Technology & Society, 2015.

6 Derrel Fincher. *Bring your own device (BYOD) programs in the classroom: Teacher use, equity, and learning tools*. PhD thesis, Pepperdine University, July 2016.

7 Karl Kapp. *The gamification of learning and instruction: Game-based methods and strategies for training and education. San Francisco, CA: Pfeiffer*. Pfeiffer, 2012.

8 A. Maia, F. Portela, and M. F. Santos. Web intelligence in higher education: A study on the usage of business intelligence techniques in education. In *2018 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pages 176–181, 2018.

9 Chirag Patel, Mahesh Gadhavi, and Atul Patel. A survey paper on e-learning based learning management systems (lms). *International Journal of Scientific and Engineering Research*, 4:171–176, June 2013.

10 Filipe Portela. A new and interactive teaching approach with gamification for motivating students in computer science classrooms. In *ICPEC 2020*, 2020.

11 Filipe Portela, Ailton Moreira da Veiga, and Manuel Filipe Santos. Benefits of Bring Your Own Device in Healthcare. In *Next-Generation Mobile and Pervasive Healthcare Solutions*, pages 32–45. IGI Global, Hershey, PA, USA, 2018. doi:10.4018/978-1-5225-2851-7.ch003.

On the Nature of Programming Exercises

Alberto Simões
2Ai, School of Technology, IPCA, Barcelos, Portugal
https://ambs.zbr.pt
asimoes@ipca.pt

Ricardo Queirós
CRACS, INESC-Porto LA, Portugal
uniMAD, ESMAD, Polythechnic of Porto, Portugal
http://www.ricardoqueiros.com
ricardoqueiros@esmad.ipp.pt

Abstract

There are countless reasons cited in scientific studies to explain the difficulties in programming learning. The reasons range from the subject's complexity, the ineffective teaching and study methods, to psychological aspects such as demotivation. Still, learning programming often boils down to practice on exercise solving. Hence, it is essential to understand that the nature of a programming exercise is an important factor for the success and consistent learning.

This paper explores different approaches on the creation of a programming exercise, starting with realizing how it is currently formalized, presented and evaluated. From there, authors suggest variations that seek to broaden the way an exercise is solved and, with this diversity, increase student engagement and learning outcome. The several types of exercises presented can use gamification techniques fostering student motivation. To contextualize the student with his peers, we finish presenting metrics that can be obtained by existing automatic assessment tools.

2012 ACM Subject Classification Applied computing → Education

Keywords and phrases Programming Exercises, Computer Science, Automatic Evaluation, Programming Challenges

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.24

Funding This work was partly founded by Portuguese national funds (PIDDAC), through the FCT – Fundação para a Ciência e Tecnologia and FCT/MCTES under the scope of the project UIDB/05549/2020.

1 Introduction

There is an unanimity regarding the difficulties founded in the teaching-learning process of computer programming. These difficulties are emphasized mainly in introductory teaching, where novice students often lack the knowledge of fundamental programming constructs. Another explanation is that students, despite being familiar with the constructs, lack the ability of "problem solving" [9]. Other studies focus on social aspects, since novice students usually have their introductory programming classes in one of the most difficult periods of their life, that is, at the beginning of a higher education course in computer science, coinciding with a period of transition and instability in their life. There are even authors who consider that the programming courses are not well located in standard computer programming degrees curricula [1, 2].

In recent years, computer programming training environments appeared with the goal of helping users to learn programming. The methodology used focus on solving problems from scratch. Nevertheless, initiating the resolution of a program can be frustrating and demotivating if the student does not know where and how to start. Based on this fact, some training environments appeared with the support for skeleton programming which facilitates

a top-down design approach, where a partially functional system with complete high-level structures is available. So, the student needs only to progressively complete or update the code to meet the requirements of the problem. Despite its promising results, there are few environments that vary their exercise types in order to motivate novice students and keep them focused.

This paper starts by presenting the life cycle of a programming exercise: how it is formalized, how it is presented to the student and how a student's solution is evaluated. Then, the study explores other ways to challenge the student avoiding the "create from scratch" assignment.

The rest of the paper is structured as follows: Section 2 explores the current state regarding programming exercise formalization and evaluation. Follows Section 3 where different approaches to construct a programming exercise are analyzed. Finally, the main contributions of the paper and possible paths for future developments are presented.

2 Programming Exercises

The way a programming exercise is formalized and evaluated is crucial for computer programming practice. In the following subsections we discuss both.

2.1 Formalization

Until two decades ago, programming assignments were created and presented to students in a *ad hoc* fashion. The increasing popularity of programming contests worldwide resulted in the creation of several contest management systems. At the same time Computer Science courses use programming exercises to encourage the practice of programming. The interoperability between these type of system is becoming a topic of interest in the scientific community. In order to address these interoperability issues several formats to represent computer science exercises were developed [6]. As notable examples we can found KATTIS [3], FreeProblemSet[1], Mooshak Exchange Format [4], PExIL [7] and YAPeXIL.

The majority of the formats, despite the syntactically differences, adhere to the same logic in terms of structure. They are based in a XML manifest file referring several types of resources such as problem statements (e.g. PDF or HTML documents), images, input/output test files, validators (static or dynamic) and solution implementations. Recently, the YAPExIL, based on PExIL, break these similarities changing the serialization format to JSON and supporting different types of programming exercises such as solution improvement, bug fix, gap filling, block sorting, and spot the bug[2].

In terms of semantics, all the formats allow the inclusion of:

- metadata: data providing information about the exercise. Usually used for discovery actions in repositories;
- instructions: text that is presented to the student (e.g., statement, instructions, skeleton code). This data is commonly presented to the student in playground (or training) environments;
- tests: data which is used by the assessment tools to evaluate the student's code. The most common data in this category is a set of tests (usually as input/output pairs) and a working solution;
- tools: tools that the author may use to generate data (e.g. feedback and tests generators, plagiarism tools).

[1] `https://github.com/davideuler/freeproblemset`
[2] These types of problems will be discussed in depth in Section 3.

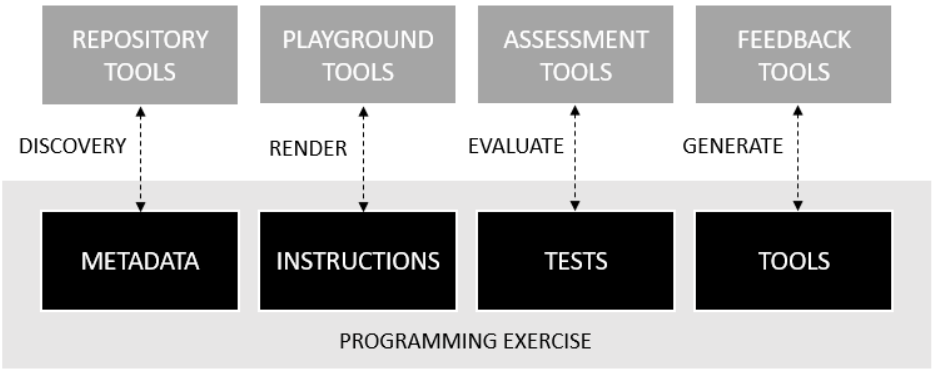


Figure 1 Anatomy of a programming exercise.

In Fig. 1 the four facets and potential tools which will consume the facet data are presented.

2.2 Evaluation

The standard way of evaluating a program is to compile it and then execute it with a set of test cases, comprised of pairs of input/output files. The submitted program is accepted if it compiles without errors and the output of each run is what is expected. This is called of dynamic evaluation.

Another approach, is using static evaluation tools that, instead of executing the student's code and injecting input data, analyzes the code and a predefined set of metrics is computed. In this context, the presence of a particular keyword or code block, the style of the code (e.g. variable naming convention), the presence of comments or even the application of a certain algorithm can be verified. For this type of analysis *linters*, or other static analysis tools are generally used.

There are several systems that fits on this category such as DOMJudge[3], Mooshak[4], PC[2][5] and DMOJ[6].

Most of these systems are contest management systems in the web. They allow the creation of creating problems, whose solutions can be written in different programming languages, and have mechanisms to judge automatically the solutions providing (web)interfaces for teams, the jury and the general public. Some of them (Mooshak and DOMJudge) also provide a REST API to allow their internal functions to be used in other scenarios. All of them are free and open source making them easy to adapt for each one needs.

3 Types of Programming Exercises

There are different approaches to create a programming exercise, depending on what is asked as task for the student, but also in the way the assignment is evaluated and graded. In this section we will first present the different ways a programming exercise can be presented to the student, and what are the main goals of that exercise, and their pros and cons. In some

[3] https://www.domjudge.org/
[4] https://mooshak.dcc.fc.up.pt/
[5] https://pc2.ecs.csus.edu/
[6] https://dmoj.ca/

cases examples of application will be discussed. It is followed by an overview of different ways an exercise can be graded, according to the objective. Finally, we will also discuss different ways to give feedback to the students about their performance.

3.1 Exercise types

While a programming exercise can be presented in very different ways, there are some that are traditional and widely used, while some other are rarely applied. These types of exercise not only present a different challenge to the student, but also can be more or less adequate for some specific type of evaluation. And, unfortunately, some types of exercises can take some time for the teacher to prepare.

Code from Scratch

This is the common approach. Easy for the teacher to prepare, as only a statement of a problem is needed. A test suite to evaluate the student's answer is needed just in the case of using an automatic evaluation tool.

For the students, they have a blank sheet, and they will need to code from scratch. In the student's point of view, this is the worst problem situation. Just like a writer or a painter, they may have the blank page syndrome. There is no indication of where to start. Students can start focusing on the main algorithm to solve the problem, but some students will start with the auxiliary/irrelevant code that is needed, and try to focus later (and probably too late) in the code that the teacher wants to evaluate.

In some situations, like when teaching Object Oriented Programming (for example using Java or C#), and particularly in the first classes, asking the student to write a static class to be able to write a static main function is counter intuitive, and breaks the Object Orientation logic.

Code Skeleton

To alleviate the blank page syndrome, and make the student focus on the piece of code being evaluated, the solution is to present a code skeleton, with some blanks to fill in. These can be simple function calls up to complete function or method bodies. Depending on the way this type of problem is presented to the student, the main part of the application might be hidden, and the student will never see the big picture. While this can seem like a bad thing, the fact that the student can focus is a great benefit. The skeleton programs will accelerate the beginning of exercises resolution and facilitate their problem understanding. With the structure included, students can now focus on the core of the problem and abstract their foundations.

As for the teacher, further work is needed. Teachers will need to write the code skeleton, and present the students with a clear interface, knowing exactly what is available at that point in the program. For complex exercises, teachers might need to write a full solution before being able to understand what pieces of code are to be developed by the student.

Fill the Blanks

Similar to the previous approach, but with smaller blank sections. Students will not need to write full lines or blocks of code, but rather fill in some portions of it. These blanks can be open, allowing the student to write whatever they want, or a predefined list, asking the students to use one of the provided options to fill the blank.

This second approach can be interesting if the students do not have the possibility to run the code, and are presented with very similar options, that will force to really understand what they are performing, without being able to test the code.

In fact, asking students to solve programming tasks without the ability to compile or run their code is relevant, as current compilation times are so fast, that students tend to try all the options/combinations possible for a specific algorithm in order to find the right answer (brute force programming).

Code Baseline

While in the previous approach the teacher will leave concrete instructions on what code needs to be written, with a code baseline, students will have access to a fully working solution. This working solution might solve the problem for specific values, and students will need to work their code to get a better solution.

This approach is very useful for (but not limited to) teaching how to implement machine learning tools. The teacher can include a solution with a precision baseline, asking the students to accomplish better.

Having a fully working solution, students feel more comfortable as they do not need to write their code from scratch, and feel empowered, as they have a working solution. Nevertheless, to start changing the code to get better results, students will need to understand the provided code, and that can be a challenging task, especially if the supplied code is not well documented, or the student is not directed to the code function he needs to change.

As a side benefit from this approach, gamification is implicit. If there is feedback on how well the student's solution is performing, they will quickly try to beat their friends solutions.

Find the Bug

In this type of exercise the student is asked to merely find the bug (or bugs) for a presented solution. These exercises are used to make students understand an algorithm logic. If the students are in a condition where they cannot compile and test the solution, this is a very interesting approach, as the student is not asked to fix the code.

Buggy Code

Students do not like to rewrite code, trying to make it faster, more elegant, bug free or more generic. The "Find the Bug" type of exercise is a good way to force students to read other's code, understand it, and change it. They are provided a buggy solution, and need to fix it.

The types of bugs introduced in the solution can be of different type accordingly with the exercise objectives:

- compilation errors: specific syntactic problems are present, like wrong variable types, missing castings, wrong function names, parameter order, etc.
- logic errors: the algorithm has serious flaws, and the student needs to detect them. If properly created, these errors can be useful to force the student to understand specific details of an algorithm.
- solution errors: the algorithm is mostly working, but have some problems in corner cases. This is similar to the "Baseline" approach, but rather than trying to raise the coverage, precision or accuracy of the solution, the student is asked to make the buggy code work on specific test cases.

Compiling Errors

With the spread of intelligent IDEs, students get used to look to the code suggestions, and very little to the compiler output. An interesting exercise to force students to look and understand how compilers analyse the code, and how they report syntax errors, is to present the student with a snippet of code with a syntax error, and the compiler message. This would be especially interesting if the code snippet is not possible to compile isolated (it uses unknown methods) and if the implementation goal is not described. This will force the student to look carefully to the error message, and to parse the code himself.

Code Interpretation

Just like reading compilation error logs, students lack the ability to understand code. A simple approach to force students to read and interpret an algorithm is to present the student with a snippet of code, and a set of options of behaviour. The behaviour can be a description of the algorithm goal, or just information about compilation error messages, or faulty behavior. This kind of exercise is interesting if the code is done in a way the student is not able to copy it and run in a compiler to test it, for example, using non defined functions described by text.

Keyword Use

This option is an add-on to some specific type of exercises, like the implementation of code from scratch or the development of a specific function or line of code. In this add-on, the teacher specifies the use of a specific keyword. As an example, the teacher may require the student to use a map function for a functional style solution to a specific problem, instead of implementing it as a loop. The main problem on this approach is the automatic evaluation, because it can not be just a pattern match, as students might use comments to put there the keyword, or include the keyword in void context, where it does not affect the behaviour of the code. Therefore, the better approach is to instrument the original function in order to log what was its input, and test there it was implemented correctly. The ability to do this kind of instrumentation will depend largely on the used programming language.

3.2 Exercise gamification modes

If different types of exercise test the students knowledge in different situations, gamification increases motivation, challenging their knowledge. Gamification can be introduced just with a ranking on the number, on how many problems were solved by each student, or by assigning (different) points to (different) problems. But this is a rather limited approach to Gamification. Gamification can be used to challenge students to solve a solution in a specific way, and therefore, being not just useful for motivation, but also for learning [8].

We will discuss how different approaches of gamification can be used to foster learning, and defy students.

Slender / Golf

Instead of just grading a solution accordingly with their result, evaluate the number of characters, instructions, or lines used to solve the problem. In some programming language communities like the Perl Community [5] this is seen like a sport, known as Golf or Golfing.

While this challenge is funny, it can be counterproductive. Shorter solutions are usually ugly, difficult to maintain, and explore obscure details of the host programming language. Therefore, while this kind of evaluation can be used with students, it should not be their main goal.

Sprinter

Efficiency is something students should understand and be able to reach. Teaching them Program Complexity is tedious and non attractive. But if students are challenged to write a fast solution for a problem, they will need to understand the efficiency of different algorithms and data structures in order to score.

If the solutions are run on a specific hardware (like a server responsible for evaluating the answers), the teacher can prepare a good solution, time it, and define a threshold execution time, forcing students to get their running time below that value.

Economic

Parallel to the Sprinter approach, students are rewarded by the amount of memory they use. Nowadays, given the large amount of memory available on personal computers, students do not have the care to use and reuse memory.

For instance, when teaching the C programming language, it is hard to teach students when they can free memory. This leads to solutions where memory is never freed. Computing the maximum amount of memory used by the solution application during a complete run can be used as a mechanism to motivate students to try to free up memory whenever they do not need it.

Sedulous

Students with learning difficulties can demotivate easily, as they see other students being able to accomplish working and probably fast and economic solutions. Rewarding students that attempt to solve a problem more than a fixed amount of tries can be motivational. Of course that the grading system should be able to understand if those are honest attempts or if the student is just trying to send always one wrong solution just to be rewarded.

Scout

Provides a bonus reward when the student makes several tests to check his solution, before submitting. This is not something that can be easily accounted for automatically. A good alternative is to give a bonus to the student if it passes all the tests with the first submitted solution.

Meticulous

Sometimes there are different ways to accomplish a working solution, and the number of lines, the code efficiency or amount of used memory is not enough to distinguish the chosen solution approach. With this in mind, teachers can define a set of specific keywords or function/methods that will give a bonus to the student's solution.

The main problem for this solution is the possibility of cheating. If the student gets aware that a specific keyword is being checked, he might just write that keyword in a comment, or in a void context, where it is not exactly being used as it should. A way to circumvent this

cheating approach is to hide to the student how the grading system works, or to define a wrapper to the functions being tested, that evaluate how they are being used in the student's solution.

3.3 Exercise Statistics

In the previous section we presented some ways to grade students accordingly with different factors that do not relate necessarily with the correctness of the solution. Teachers might not want to use all of those grading approaches at the same time. Nevertheless, computing statistics on some of the presented factors, and showing them to the students can work, indirectly, as gamification.

Thus, it is suggested to add solution metrics regarding each problem submitted solutions. Follow some simple examples:

- Average Solution Time: how much time a student takes to prepare a solution, starting from the moment the problem description was seen, up to a good solution to be submitted. This will allow students to understand how they relate with their mates, and will allow the teacher to understand how his students problem solving abilities are.
- Wrong Attempts Average: how many attempts (in average) a student performs, before getting the solution accepted. If this number is high, students might not have understood the problem correctly, or they are trying at force to get a solution, instead of really thinking in a good approach.
- Least Memory Used: who is the student having the solution using less memory for each problem.
- Shortest Execution Time: who is the student having the fastest solution.
- Average Execution Time: what is the average execution time for a specific problem solutions.

4 Conclusions

Learning programming is a difficult task. Many reasons have been shared among the scientific community. However, it is important not to forget, that learning programming requires constant practice. In programming, this practice boils down to solving exercises, often from scratch. While this could be simple for average and expert students, for novice students this approach can negatively affect his performance in the course and, consequently, increase their demotivation. Therefore, this paper describes several types of exercises in order to cover different learning profiles and enhance new skills. This diversity is seen by the authors as beneficial to not making the challenges tedious for more advanced students and to support novice students to consolidate their skills.

As future work, the authors will try to explore simple ways to facilitate the process of changing an exercise type through standard and non-language dependent techniques.

References

1 Anabela Gomes, Cristiana Areias, Joana Henriques, and António José Nunes Mendes. Aprendizagem de programação de computadores: dificuldades e ferramentas de suporte. *Revista Portuguesa de Pedagogia*, 42:161–179, 2008.

2 Tony Jenkins. On the difficulty of learning to program. In *3rd Annual LTSN-ICS Conference*, pages 53–58, 2002.

3 Kattis. Kattis, 2019. accessed on Jan 2020. URL: `https://open.kattis.com/`.

4 José Paulo Leal and Fernando Silva. Mooshak: A web-based multi-site programming contest system. *Softw. Pract. Exper.*, 33(6):567–581, May 2003. doi:10.1002/spe.522.

5 Jon Orwant. *Games Diversions & Perl Culture: Best of the Perl Journal*. O'Reilly Media, 2004.

6 Ricardo Queirós and José Paulo Leal. Babelo - an extensible converter of programming exercises formats. *TLT*, 6(1):38–45, 2013. doi:10.1109/TLT.2012.21.

7 Ricado Queirós and Jose Paulo Leal. Making programming exercises interoperable with PExIL. In Jose Carlos Ramalho, Alberto Simões, and Ricardo Queirós, editors, *Innovations in XML Applications and Metadata Management: Advancing Technologies*, pages 38–56. IGI Global, 2013.

8 Jakub Swacha, Ricardo Queirós, José Carlos Paiva, and José Paulo Leal. Defining requirements for a gamified programming exercises format. In Imre J. Rudas, János Csirik, Carlos Toro, János Botzheim, Robert J. Howlett, and Lakhmi C. Jain, editors, *Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES-2019, Budapest, Hungary, 4-6 September 2019*, volume 159 of *Procedia Computer Science*, pages 2502–2511. Elsevier, 2019. doi:10.1016/j.procs.2019.09.425.

9 Jacqueline L. Whalley, Raymond Lister, Errol Thompson, Tony Clear, Phil Robbins, P. K. Ajith Kumar, and Christine Prasad. An australasian study of reading and comprehension skills in novice programmers, using the bloom and solo taxonomies. In *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52*, ACE '06, page 243–252, AUS, 2006. Australian Computer Society, Inc.

Polish Python: A Short Report from a Short Experiment

Jakub Swacha
Department of IT in Management, University of Szczecin, Poland
jakub.swacha@usz.edu.pl

Abstract

Using a programming language based on English can pose an obstacle for learning programming, especially at its early stage, for students who do not understand English. In this paper, however, we report on an experiment in which higher-education students who have some knowledge of both Python and English were asked to solve programming exercises in a Polish-language-based version of Python. The results of the survey performed after the experiment indicate that even among the students who both know English and learned the original Python language, there is a group of students who appreciate the advantages of the translated version.

2012 ACM Subject Classification Social and professional topics → Computing education; Software and its engineering → General programming languages

Keywords and phrases programming language education, programming language localization, programming language translation, programming language vocabulary

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.25

1 Introduction

As a result of the overwhelming contribution of English-speaking researchers to the conception and development of computer science, almost every popular programming language used nowadays has a vocabulary based on this language [15]. This can be seen as an obstacle for learning programming, especially at its early stage, for students whose native language is not English [11]. In their case, the difficulty of understanding programs is augmented by the fact that keywords and standard library function names mean nothing to them. Even in the case of students who speak English as learned language, they are additionally burdened with translating the words to the language in which they think.

In order to let everyone write programs with the same level of ease as native English speakers can do, one needs to provide programming languages based on various natural languages. Although there were numerous attempts to develop new localized programming languages (see e.g. Algorithmi [9] or Phoenix [1]) and to localize existing languages (such as Logo as exemplified by various language kits for MSWLogo [8]), so far none of them gained notable popularity, maybe with the sole exception of Scratch [12], ranked 25th in the current TIOBE index [15].

In this paper, we investigate an attempt of translating one of the most popular programming languages of these days, Python, to the Polish language, and report the reaction of higher-education students who were asked to use it to solve just two programming exercises. As the experiment lasted short, and the survey answers obtained from the involved students were not many, the results presented here can only be interpreted to a limited degree, but they can certainly be seen as an indicator of the students' interest in non-English-based, translated programming languages, and possibly a motivator for future work on them.

The remainder of this paper is organized as follows. Section 2 gives a glimpse of the historical and still existing non-English programming languages with a focus on Polish-based languages. Next, the assumptions and the form of the experiment involving the students are

First International Computer Programming Education Conference (ICPEC 2020).
Editors: Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões; Article No. 25; pp. 25:1–25:6

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

described, along with the unsophisticated technique used to translate Python vocabulary to Polish language and the scope of translation. Then the survey results are presented and discussed. The final section concludes.

2 Related Work

The benefits of writing programs using a language based on the programmer's native language are especially valuable in the case of natural languages based on non-Latin script, where programmers can struggle even with deciphering characters they are not accustomed to. A number of programming languages were developed based on natural languages using such scripts – the examples from different computing epochs are Russian Rapira [7], developed in the 1980s for educational purposes, Chinese Eyuyan, first released in early 2000s, and still active [6], or Arabic Phoenix, released only as recently as 2019 [1].

While the benefit of using alphabet familiar to the programmer does not apply to the Polish language which is based on Latin script (with some additions), the argument of using words of a language known to the programmer remains valid. It was even more valid in the times of communist rule in Poland (1945-1989), when foreign language education in most primary schools was limited to the Russian language, and only a very small part of the population had any knowledge of English. Probably the earliest programming language based on Polish was SAKO, developed for the first Polish-built computers of the late 1950's and the early 1960's [17]. Over twenty years later, when the 8-bit microcomputers found their way into Polish homes, the Logo programming language has been translated, first for Atari XL/XE [16], then for Elwro 800 Junior, a Polish ZX Spectrum clone [3]. Contemporarily, Logomocja, a Polish version of Imagine Logo is still used for educational purposes [4].

In 2008, Rey, an educational programming language using Polish words was released. It was based on Java syntax and implemented many modern object-oriented-programming language features [2]. Unfortunately, Rey did not manage to reach a wide audience, which was, however, achieved by another educational language: Scratch, translated to Polish by Weronika Łabaj, Jan Baryła, Kris Kopera, Aleksandra Kopczynska, Marek Nowicki and Tomasz Ho-Janecki [10].

Probably the most interesting approach to the problem of programming language localization is the one introduced in Citrine, version 0.7, whose vocabulary is automatically translated between natural languages [5].

3 The experiment

3.1 Motivation

While the advantages of using a programming language similar to programmers' native language are easy to defend in the case of young pupils having no prior programming experience and limited or no knowledge of the base natural language, it is far from obvious if it could be considered valuable by higher-education students, who not only know the base natural language but also mastered the translated programming language at least at the pre-intermediate level. Note that a positive answer to this question makes it worthwhile to develop localized programming languages aiming at something more than introductory programming education, possibly even professional programming.

While a definite answer to this question is beyond the scope of this author's intended effort, as it would require a fully-fledged translation of a programming language and a long-term observation involving a significant number of students, a preliminary answer could be obtained in a very simple way as described below.

3.2 Execution

The experiment was performed at the end of winter semester of academic year 2019/2020. Two simple programming exercises were first prepared. The first exercise addressed the topics of text input/output and loop control and the second one –defining classes and methods. The exemplary solutions for the two exercises in Polish Python are presented in Listings 1 and 2.

Exercise 1. Write a program that asks the user to enter a password and checks whether it is `Mniam!`. If the password is wrong, the user is asked to correct it, but if he/she fails for the third time, the program should close.

Listing 1 Exemplary solution for Exercise 1.

```
dla _ w zakres(3):
    kod = wczytaj('Podaj tajny kod obiadowy: ')
    gdy kod == 'Mniam!':
        pisz ('To jest dobre hasło!')
        przerwij
    inaczej:
        pisz ('To nie jest dobre hasło!')
inaczej:
    pisz ('Wyczerpano dostępną liczbę prób.')
```

Exercise 2. Write a `Square` class which features one field (side length) and two methods (area and perimeter). Create an object of this class and call both its methods.

Listing 2 Exemplary solution for Exercise 2.

```
klasa Kwadrat:
    a = 1.0
    funkcja obwod(f):
        powrót f.a*4
    funkcja pole(f):
        powrót f.a**2
k = Kwadrat()
pisz (k.obwod(), k.pole())
```

Three groups counting together over 80 students were asked to read a short introduction to the translated Python language including a list of words translated to Polish (see Table 1), then solve the exercises using the translated version of the language. The list of translated words comprised all the Python keywords, most built-in functions as well as several module functions and methods that were chosen considering the typical solutions of the exercises used in the experiment. In order to edit and test their solutions, the students were given access to a modified version of this author's web-based interactive learning environment for Python described in [14]. The modification comprised in using a simple RegExp to replace all occurrences of the original Python words listed in Table 1 with their respective Polish translations.

Eventually, having successfully finished the exercises, they were asked to answer a survey consisting of just four questions:

1. General attitude to Python featuring Polish words.
2. The effect of translation on code readability.
3. The perceived value of translation for learning programming.
4. The willingness to write longer programs using the translated language.

The whole experiment took less than an hour.

Table 1 The scope of translation: Python words translated to Polish.

Original	Translated	Original	Translated	Original	Translated
add	dodaj	and	i	append	dostaw
as	jako	assert	sprawdź	await	oczekuj
break	przerwij	capitalize	dużą	ceil	sufit
center	centruj	choice	wybierz_losowo	chr	znak
class	klasa	clear	czyść	continue	kontynuuj
copy	kopiuj	count	licz	def	funkcja
degrees	stopnie	del	skasuj	dict	słownik
difference	różnica	discard	wyrzuć	elif	ale gdy
else	inaczej	except	wyjątek	extend	wydłuż
False	Fałsz	finally	finalnie	find	znajdź
float	dziesiętna	floor	podłoga	for	dla
from	z	frozen_set	stały_zbiór	global	globalne
if	gdy	in	w	index	pozycja
input	wczytaj	insert	wstaw	int	liczba
intersection	przecięcie	is	to	join	złącz
keys	klucze	len	długość	list	lista
lower	małe	math	matematyka	None	Nic
nonlocal	nielokalne	not	nie	or	lub
ord	kod	pass	pas	print	pisz
radians	radiany	raise	podnieś	randint	liczba_losowa
random	losowe	range	zakres	raw_input	wczytaj_tekst
remove	usuń	replace	podmień	return	powrót
reverse	odwróć	rfind	znajdź_od_tyłu	rjust	do_prawej
round	zaokrągl	set	zbiór	setdefault	ustaw_ domyślną
shuffle	pomieszaj	sort	sortuj	sorted	posortowana
split	podziel	sqrt	pierwiastek	str	napis
sum	suma	swapcase	zamień_litery	symmetric_ difference	różnica_ symetryczna
title	tytuł	True	Prawda	try	spróbuj
tuple	krotka	type	typ	union	połączenie
upper	duże	values	wartości	while	dopóki
with	używając	yield	zwróć	___init___	___inicjuj___
___str___	___napis___				

4 Survey results

The participation in the survey was not compulsory and some of the students were busy working on their assignments behind time. Altogether 48 survey responses were received (one omitting the answer to the first question).

For question 1, the allowed answers ranged from 1 (Nonsense) to 5 (Highly interesting). The distribution of students' answers to this question is presented in Fig. 1.

While about half of the surveyed students evaluated the idea negatively, there is about one-third of them who considered such an approach as interesting.

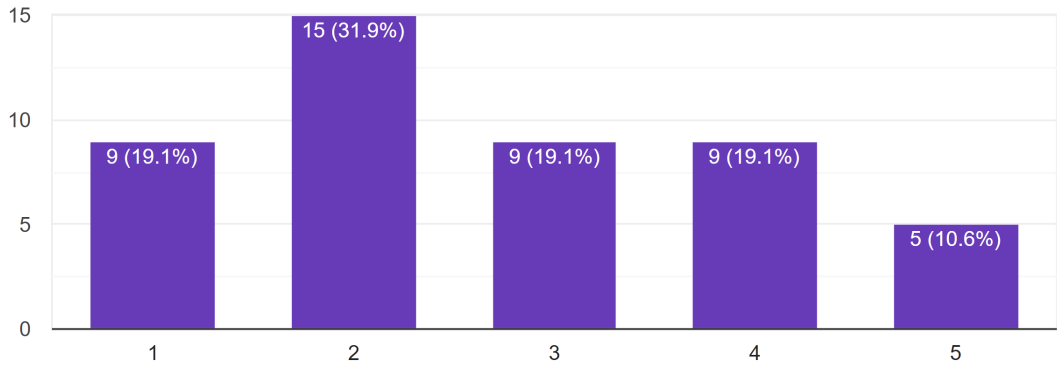


Figure 1 Overall evaluation of the idea of translating Python to Polish.

For question 2, about half of the students declared the effect of translation on code readability as negative, whereas only about 15% considered it positive. In the context that the students were accustomed to the standard, English-based version of Python, it is worth noting that there was a non-marginal group of students who appreciated the translation to their native language.

With regard to question 3, almost one-third of the students agreed with the benefits of translation for introductory programming learning. About one-sixth considered it as pointless because students should know English. According to half of the respondents the downside is that it would make it difficult to switch to the standard, English-based Python at the later stage of learning.

Finally, four-fifths of the students would not like to write any longer program using the translated Python, the remaining 20% of the surveyed students had opposite opinion.

5 Conclusion

The experiment described in this paper revealed that even among the higher-education students with prior experience with at least one English-based programming language, and some command of English, there is a group which finds the translated version of program more readable. This provides some motivation to future work on non-English-based, translated programming languages.

While the technical approach used in the experiment (using a simple RegExp) shows how easy it is to implement program translation in the world of browser-accessed interpreters, applying the translation to a stand-alone interpreter would require a more sophisticated approach, though there are ready-made solutions for doing it (see e.g. [13]).

One problem that would be difficult to tackle in real world is the scope of translation. Leaving popular libraries untranslated would leave the code look only partly-translated. On the other hand, in practice, it is impossible to translate them wholly, considering that they keep being developed and updated every day. Perhaps, the solution could be automatic translation, though it is doubtful whether its reliability matches the strict translation requirements necessary for the translated programs to run correctly.

References

1. Youssef Bassil. Phoenix – the Arabic object-oriented programming language. *International Journal of Computer Trends and Technology*, 67(2):7–11, 2019.
2. Paweł Baszuro and Jakub Swacha. Rey: An educational programming language. In R.T. Mittermeir and M.M. Sysło, editors, *Informatics education: Contributing across the curriculum*, pages 1–9, Toruń, 2008. WMiI UMK/PTI.
3. Wojciech Cellary and Krzysztof Pielesiak. *Leksykon Logo*. WNT, Warsaw, 1990.
4. Wojciech Czerski. Nowe sposoby nauki programowania w edukacji wczesnoszkolnej. *Dydaktyka Informatyki*, 13:129–134, 2018. doi:10.15584/di.2018.13.16.
5. Gabor de Mooij. Citrine, 2020. accessed on 20 Jan 2020. URL: http://citrine-lang.org.
6. DYWT.COM.CN. Eyuyan 5.9, 2019. accessed on 20 Jan 2020. URL: http://www.eyuyan.com/pdown.htm.
7. Laura Eileen Goodin. Teaching a nation to compute: The Rapira project and Soviet Information Technology education. Master's thesis, The American University, Washington, 1991.
8. Brian Harvey and George Mills. MSWLogo, an educational programming language, 2020. accessed on 20 Jan 2020. URL: http://www.softronix.com/logo.html.
9. António Manso, Luís Lopes, Célio Gonçalo Marques, Raquel Guedes, and Paulo Santos. Algorithmi: Bridging the algorithms to natural and programming languages, 2020. accessed on 20 Jan 2020. URL: http://algorithmi.ipt.pt/assets/papers/Final_Paperalgorithmi_2019_versao_EN.pdf.
10. MIT Scratch Team. Translators, 2020. accessed on 20 Jan 2020. URL: http://en.scratch-wiki.info/wiki/Translators.
11. Yizhou Qian and James D. Lehman. Correlates of success in introductory programming: A study with middle school students. *Journal of Education and Learning*, 5(2), 2016. doi:10.5539/jel.v5n2p73.
12. Mitchel Resnick. Reviving Papert's dream. *Educational Technology*, 52(4):42–46, 2012.
13. Stack Overflow. Can you add new statements to Python's syntax?, 2018. accessed on 20 Jan 2020. URL: http://stackoverflow.com/questions/214881/can-you-add-new-statements-to-pythons-syntax.
14. Jakub Swacha. Development and evaluation of an interactive Python course. In *ICERI2018 Proceedings*, pages 456–466, Seville, 2018. IATED.
15. TIOBE. TIOBE Index for January 2020, 2020. accessed on 20 Jan 2020. URL: http://www.tiobe.com/tiobe-index/.
16. Wojciech Zientara. Polskie Logo, 1986. accessed on 20 Jan 2020. URL: http://archive.org/details/a8b_Polskie_Logo_Wersja_Kasetowa_1986_W.Zientara_of_Bajtek_pl.
17. Leon Łukaszewicz and Antoni Mazurkiewicz. SAKO – an automatic coding system. *Annual Review in Automatic Programming*, 2:161–176, 1961.

A Roadmap to Gamify Programming Education

Jakub Swacha
University of Szczecin, Poland
jakub.swacha@usz.edu.pl

Ricardo Queirós
CRACS – INESC-Porto LA, Portugal
uniMAD, ESMAD/Polythechnic of Porto, Portugal
ricardoqueiros@esmad.ipp.pt

José Carlos Paiva
CRACS – INESC-Porto LA, Portugal
DCC – FCUP, Porto, Portugal
up201200272@fc.up.pt

José Paulo Leal
CRACS – INESC-Porto LA, Portugal
DCC – FCUP, Porto, Portugal
zp@dcc.fc.up.pt

Sokol Kosta
Aalborg Universitet, Denmark
sok@es.aau.dk

Raffaele Montella
Università Degli Studi Di Napoli "Parthenope", Italy
raffaele.montella@uniparthenope.it

Abstract

Learning programming relies on practicing it which is often hampered by the barrier of difficulty. The combined use of automated assessment, which provides fast feedback to the students experimenting with their code, and gamification, which provides additional motivation for the students to intensify their learning effort, can help pass the barrier of difficulty in learning programming. In such environment, students keep receiving the relevant feedback no matter how many times they try (thanks to automated assessment), and their engagement is retained (thanks to gamification).

While there is a number of open software and programming exercise collections supporting automated assessment, up to this date, there are no available open collections of gamified programming exercises, no open interactive programming learning environment that would support such exercises, and even no open standard for the representation of such exercises so that they could be developed in different educational institutions and shared among them. This gap is addressed by Framework for Gamified Programming Education (FGPE), an international project whose primary objective is to provide necessary prerequisites for the application of gamification to programming education, including a dedicated gamification scheme, a gamified exercise format and exercises conforming to it, software for editing the exercises and an interactive learning environment capable of presenting them to students. This paper presents the FGPE project, its architecture and main components, as well as the results achieved so far.

2012 ACM Subject Classification Social and professional topics → Computer science education

Keywords and phrases gamification, programming, learning, automatic assessment, programming exercises

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.26

Funding This paper is based on the work done within the Framework for Gamified Programming Education project supported by the European Union's Erasmus Plus programme (agreement no. 2018-1-PL01-KA203-050803).

First International Computer Programming Education Conference (ICPEC 2020).
Editors: Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões; Article No. 26; pp. 26:1–26:7

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Skilled programmers are in high demand in the European Union countries, and the EU has been focusing on different initiatives to increase the number of such experts, calling coding *the 21st century skill* [5]. One of the key obstacles to satisfy this demand is the difficulty in learning programming [3]. We believe a progress in this area can be attained with the combined use of automated assessment, which provides fast feedback to the students experimenting with their code, and gamification, which provides additional motivation for the students to intensify their learning effort. In our opinion, the availability of programming courses based on such an approach may not only improve the effectiveness of learning programming, but also extend the group of people feeling capable of effectively learning it. Nonetheless, to the best of the authors' knowledge, there are yet no available open collections of gamified programming exercises, no open interactive programming learning environment that would support such exercises, and even no open standard for the representation of such exercises, so that they could be developed in different educational institutions and shared among them.

In this paper, we present a work-in-progress on a framework for application of gamification to programming education, realized as an international project within the scope of the Erasmus+ programme, key action: *Cooperation for innovation and the exchange of good practices* [7]. The scope of this project includes the specification of the gamification scheme and the exercise definition format, and the development of a collection of gamified exercises for several popular programming languages and software: a toolkit for editing the exercises and an interactive learning environment so that they could be given to the students. The target group of the project are programming instructors and students learning programming (also self-teaching). All the project outputs will be freely available on the Internet under open-source licenses. The expected impact of the project is an improvement in the efficiency of programming education and its student-perceived experience.

The remainder of this paper is organized as follows. Section 2 presents an overview of the existing private platforms and open source tools related to gamified programming. Then, Section 3 provides a brief insight of the proposed framework, including details on its architecture, design, and implementation. Also, it enumerates and describes the intellectual outputs (IOs) of the project and their current status. Following, Section 4 presents the first results achieved by the project team. Finally, Section 5 summarizes the main contributions of this research.

2 Related Work

Many companies and educational institutions are investing in solutions to teach programming to young students. A good example of this is the European Coding Initiative [11], a project supported by the European Commission that aims at *promoting coding and computational thinking at all levels of education.* This initiative provides a series of documents and guidelines for teachers on how to teach programming at different student levels and for students on how to practice and learn coding. Google has introduced the *Code with Google* initiative [8], where students can attend remote coding classes and work in teams to learn programming. Moreover, Google has different competitive programming contests, where coders solve a set of problems in a certain amount of time and they get ranked based on their performance [9].

Similar to Google's Coding Competitions, many other online tools exist, such as HackerRank [10], TopCoder [20], LeetCode [14], among others. They all present students with hundreds or thousands of problems and rank them using several metrics, such as number of

problems solved, total points collected by solving each problem, efficiency of the solution, among others. Quite often, recruiters use these platforms to identify skilled programmers to offer them a chance for a job interview [1]. However, these successful platforms are owned and managed by private companies, meaning that there is no collaboration among them. One direct consequence of this factor is that the same programming problems are being rephrased and entered multiple times in the different platforms.

Open-source solutions for competitive programming contests exist and have been adopted by different universities to teach coding. Some of the most prominent ones are Kattis [12], DOMjudge [6], DMOJ [4], and Mooshak [13]. Being open-source, educators can install these tools in their servers and provide a means for students to compete with each other. Unfortunately, each tool uses a self-established format for storing exercises [15, 16, 13, 12, 18], without adhering to a common format, which hinders the sharing of these problems. As a consequence, an open collection of problems is not yet available to educators, so they will have to spend their time creating the problems in the tool of their choice.

In addition to that, competition is the only motivational element typically included with automated assessment tools out-of-the-box. However, too much focus on competition can be harmful to those who lose [21], which is commonly the case of students with learning difficulties. For instance, gamification, which includes but is not limited to competition, is an area of great interest towards the engagement of students in (programming) education [2]. Notwithstanding, there is a complete lack of formats/specifications for its application in educational contexts, increasing the disparity of implementations.

3 Framework Architecture and Planned Outcomes

The five IOs planned for the FGPE project are the following: (IO1) Gamification Scheme for Programming Exercises; (IO2) Data Exchange Format for Gamified Programming Exercises; (IO3) Tools Supporting Editing and Conversion of Programming Exercises; (IO4) Programming Courses featuring Gamified Exercises; and (IO5) Programming Learning Environment featuring Gamified Exercises.

In Fig. 1, we present the conceptual architecture of the framework based on these outputs and their relations.

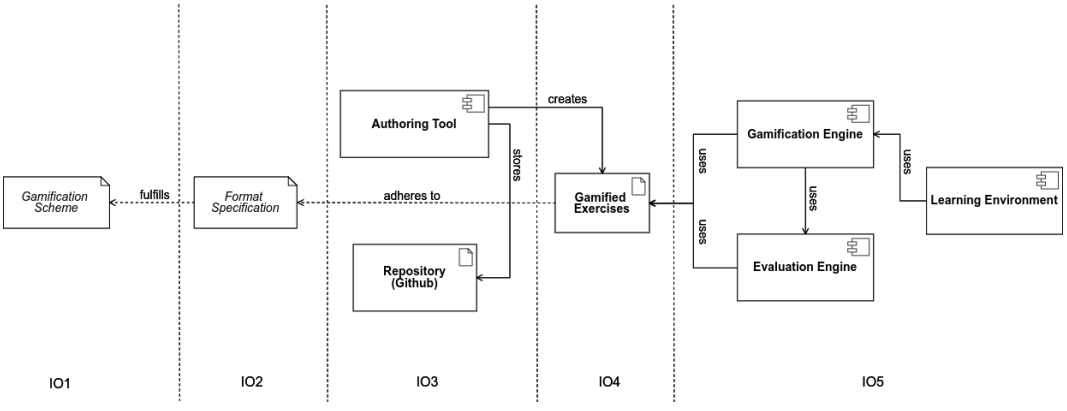


Figure 1 Conceptual architecture of the framework.

The IO1 – Gamification Scheme for Programming Exercises – aims to provide easy-to-follow guidelines on how to apply gamification to programming courses. The expected result of this IO is a reference scheme for gamification of programming courses.

In IO2 – Data Exchange Format for Gamified Programming Exercises –, the goal is to foster data reusability and interoperability in a gamified programming education framework, more precisely: programming exercises' data and gamification-related data. This will be achieved through the definition of specification formats that fulfil the gamification scheme created in the previous IO.

The IO3 – Tools Supporting Editing and Conversion of Programming Exercises – aims to simplify both the process of authoring and storing exercises and the binding of the gamification layer to a set of exercises. In order to accomplish these goals, a tool to author content adhering to the formats designed in IO2 will be created. This tool will integrate with Github to store files and data.

The IO4 – Programming Courses featuring Gamified Exercises – will provide a base set of ready-to-use exercises, created with the tools developed in IO3 and conforming to the formats defined in IO2, that addresses the needs of the most popular programming courses taught by the project partners.

Finally, the IO5 – Programming Learning Environment featuring Gamified Exercises – consists of the development of a free and open-source learning environment that will use the gamification and the evaluation engines to engage and assess students' performance. This learning environment will consume the exercises developed at IO4.

4 First results

Currently, the first three IOs are completed. The remaining two are still in progress, and not yet suitable to be presented.

The first IO involved the development of the reference scheme for gamification of programming courses. It resulted in the identification of gamification concepts and techniques applying them to programming education, as well as the use cases for gamified programming exercises and their matching to the respective gamification techniques. The results of this study were validated with a survey administered to students of five distinct universities, and described in a previous publication [17].

Having selected the gamification concepts of interest for programming education, a study to gather the requirements bound to those concepts for the specification of a gamified exercise format has been conducted [19]. As a preliminary result, this study unveils the importance, for reusability and interoperability, of separating the two main kinds of data present in a gamified programming education framework: programming exercises' data and gamification-related data. The former includes data relating to the full life-cycle (creation, selection, presentation, solving, and evaluation) of a programming exercise, and even though a standard format or, at least, a consensus is yet to be found, it is well-grounded in the literature due to its extensive use by automated assessment tools [15, 16, 13, 12, 18]. The latter conveys all the elements that should be included to foster the motivation and enjoyment of students during the realization of the programming exercises (e.g., challenges and rewards), and is the novelty of the study as there is a complete absence of formats for its application in education (not only programming education).

This distinction led to the development of two independent formats for IO2, namely YAPExIL and GEdIL, as well as a tool to support the authoring of content adhering to these formats, the AuthorKit, for IO3.

4.1 YAPExIL

Yet **A**nother **P**rogramming **Ex**ercises **I**nteroperability **L**anguage (YAPExIL) is a JSON schema for describing a programming exercise package, based on PExIL [15] – an XML dialect that aims to consolidate all the data required in the programming exercise life-cycle. YAPExIL aims for simplicity while covering most features of a task package as described by Verhoeff's model [22], i.e., a unit for collecting, storing, archiving, and exchanging all information concerning with a programming task.

YAPExIL is composed of four facets: metadata, presentation, evaluation, and tools. Metadata facet contains identification information about an exercise, such as title, author, keywords, module in which it is contained, among others. Presentation facet includes what is presented to the student when he/she opens the exercise as well as instructions displayed to teachers for reusing exercises. Evaluation facet encompasses all the components used in the automated assessment of the exercise. Finally, the tools facet holds extra external scripts that are not strictly necessary at any phase of the programming exercise life-cycle.

4.2 GEdIL

Gamified **Ed**ucation **I**nteroperability **L**anguage (GEdIL) is a JSON schema designed to describe gamification layers for educational contexts. Although designed to fulfil the specific requirements of gamification applied in programming courses [19], GEdIL is sufficiently generic to be applied to any other educational subjects as it only delineates a layer with the gamification elements that should lay on top of another layer describing activities.

GEdIL defines a hierarchy of challenges where each level (including the root) may have rules, rewards, and leaderboards. The root node has metadata for identifying the layer. Leaf nodes refer to activities, bridging to bottom layers. In this way, the gamification layers are easily replaceable and may reuse the same activities.

4.3 AuthorKit

AuthorKit is a web application to author educational content adhering to the formats of the FGPE (YAPExIL and GEdIL). Each author can create/edit/view educational content inside projects he/she owns and projects shared with him/her by other authors.

The user interface, presented in Fig. 2, is characterized by two separate form wizards nested under a project: one to create programming exercises and another to create gamification layers. The former maps each facet of YAPExIL to a step of the wizard, having the possibility to upload all the necessary files for a programming exercise (e.g., problem statement, usage instructions, input/output files for tests, and solutions). The latter has steps to (1) manage the metadata of the gamification layer, (2) add rewards that are not linked to a challenge, (3) wire the rules of the course, (4) create the leaderboards by defining their metrics, and (5) build the challenge tree, where each challenge may have child challenges and/or local scoped rules, leaderboards, and rewards. All data, including files, is synchronized to a repository within the GitHub account linked to the project owner's account of AuthorKit.

5 Conclusion

In this paper, we presented our work-in-progress in the area of programming education, particularly targeting the aspect of student engagement. Our contribution will advance the state of the art by addressing the gap of the lack of open collections of reusable gamified programming exercises by providing: *(1)* a frame of reference for programming

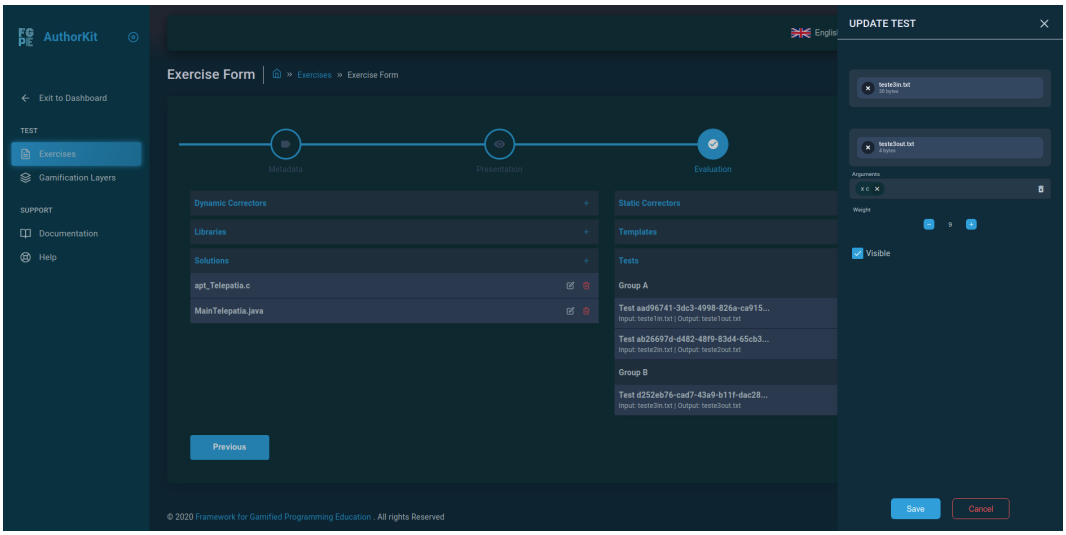


Figure 2 Editing a test on the Evaluation step of the exercise form wizard (dark theme).

course gamification (including featured gamification concepts and the intended area of their application); *(2)* the specification of a format for exchanging gamified programming exercises based on the above frame of reference; *(3)* tools for authoring exercises in the above format; *(4)* a programming learning environment allowing to set up and manage gamified programming courses making use of such exercises. These four components constitute the scope of the Framework for Gamified Programming Education project [7].

References

1 Bharat Adibhatla. Top 5 coding websites companies hire from, 2019. accessed on 20 Jan 2020. URL: `https://analyticsindiamag.com/top-5-coding-websites-companies-hire-from/`.

2 Manal M. Alhammad and Ana M. Moreno. Gamification in software engineering education: A systematic mapping. *Journal of Systems and Software*, 141:131–150, 2018. `doi:10.1016/j.jss.2018.03.065`.

3 Yorah Bosse and Marco Aurélio Gerosa. Why is programming so difficult to learn? patterns of difficulties related to programming learning mid-stage. *SIGSOFT Softw. Eng. Notes*, 41(6):1–6, January 2017. `doi:10.1145/3011286.3011301`.

4 Guanzhong Chen. Dmoj, 2020. accessed on 20 Jan 2020. URL: `https://dmoj.ca/`.

5 European Commission. Coding - the 21st century skill, 2018. accessed on 20 Jan 2020. URL: `https://ec.europa.eu/digital-single-market/en/coding-21st-century-skill`.

6 Jaap Eldering, Nicky Gerritsen, Keith Johnson, Thijs Kinkhorst, and Tobias Werth. Domjudge, 2020. accessed on 20 Jan 2020. URL: `https://www.domjudge.org/`.

7 FGPE Project Consortium. Framework for gamified programming education, 2018. accessed on 20 Jan 2020. URL: `http://fgpe.usz.edu.pl`.

8 Google. Code with google, 2020. accessed on 20 Jan 2020. URL: `https://edu.google.com/code-with-google/`.

9 Google. Google's coding competitions, 2020. accessed on 20 Jan 2020. URL: `https://codingcompetitions.withgoogle.com/`.

10 HackerRank. Hackerrank, 2011. accessed on 20 Jan 2020. URL: `https://www.hackerrank.com/`.

11 European Coding Initiative. All you need is code, 2020. accessed on 20 Jan 2020. URL: `http://www.allyouneediscode.eu`.

12 Kattis. Kattis, 2019. accessed on Jan 2020. URL: https://open.kattis.com/.

13 José Paulo Leal. Mooshak, 2018. accessed on Jan 2020. URL: https://mooshak.dcc.fc.up.pt/.

14 LeetCode. Leetcode, 2015. accessed on 20 Jan 2020. URL: https://leetcode.com/.

15 Ricardo Queirós and José Paulo Leal. PExIL: Programming exercises interoperability language. In *Conferência Nacional XATA: XML, aplicações e tecnologias associadas, 9. ª*, pages 37–48. ESEIG, 2011.

16 Erik Scheffers, Tom Verhoeff, Stefan Geuns, Marijn Kruisselbrink, Paul Wagener, Robert Leenders, Iosif Macesanu, and Maikel Steneker. peach3, 2017. accessed on Jan 2020. URL: https://peach3.nl.

17 J. Swacha, R. Queirós, and J. C. Paiva. Towards a framework for gamified programming education. In *2019 International Symposium on Educational Technology (ISET)*, pages 144–149, July 2019. doi:10.1109/ISET.2019.00038.

18 Jakub Swacha. SIPE: A Domain-Specific Language for Specifying Interactive Programming Exercises. In *Towards a Synergistic Combination of Research and Practice in Software Engineering*, pages 15–29, Cham, Switzerland, 2018. Springer.

19 Jakub Swacha, Ricardo Queirós, José Carlos Paiva, and José Paulo Leal. Defining requirements for a gamified programming exercises format. *Procedia Computer Science*, 159:2502–2511, 2019. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES2019. doi:10.1016/j.procs.2019.09.425.

20 TopCoder. Topcoder, 2001. accessed on 20 Jan 2020. URL: https://www.topcoder.com/.

21 Maarten Vansteenkiste and Edward L. Deci. Competitively contingent rewards and intrinsic motivation: Can losers remain motivated? *Motivation and emotion*, 27(4):273–299, 2003. doi:10.1023/A:1026259005264.

22 Tom Verhoeff. Programming task packages: Peach exchange format. *International Journal Olympiads In Informatics*, 2:192–207, 2008.

Improving Game-Based Learning Experience Through Game Appropriation

Salete Teixeira
Centro ALGORITMI, Universidade do Minho, Braga, Portugal
salete.teixeira97@gmail.com

Diana Barbosa
Centro ALGORITMI, Universidade do Minho, Braga, Portugal
a78679@alunos.uminho.pt

Cristiana Araújo
Centro ALGORITMI, Universidade do Minho, Braga, Portugal
decristianaaraujo@hotmail.com

Pedro R. Henriques
Centro ALGORITMI, Universidade do Minho, Braga, Portugal
prh@di.uminho.pt

Abstract

Computational Thinking is an essential concept in this technological age. Several countries have included this subject as part of their educational program, and many others intend to do it. However, this is not a regular subject like maths or history; it needs more training (to increase the capabilities/skills) than studying and memorizing concepts. So it comes clear that the introduction of Computational Thinking to students requires the choice of the most suitable learning resources. Game-Based Learning was proven to be an effective teaching method. Therefore, we elected games as our learning resources. Nonetheless, we believe that the learning experience and motivation of students when playing games can be improved by choosing the most suitable game for each student. So, this paper focuses on the adaptation of Game-Based Learning to each student to develop Computational Thinking. We will argue that this adaptation can be done in a computer supported systematic way. To make that possible, on one hand, it will be necessary to classify games – an original ontology was used for that. On the other hand, it is crucial to establish the students' profile, having into consideration sociodemographic factors, personality, level of education, among others. Then, resorting to a similarity evaluation process it is feasible to choose the games that best fit the players, augmenting the effectiveness of the learning experience. We intend to start applying our approach – training Computational Thinking – to young students, since the first scholar years. However, we are also considering its application to adults starting programming studies.

2012 ACM Subject Classification Applied computing → Education

Keywords and phrases Computational Thinking, Computing Education, Game-Based Learning, Game Types, Ontology, Student Profile, Adult Learning

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.27

Funding This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020.

1 Introduction

With the advances in technology in the past years, Computing become an area of great interest and present in almost every day life activity and jobs. Consequently, an increasing number of students choose to access Computer Science courses, and many professionals are changing careers or starting a late education in technological areas. With this popularity and being programming a field in which students present several difficulties, it emerged the

need to explore the education of Computational Thinking. Multiple researchers believe there is a gap in the school systems that should be filled with the development of Computational Thinking, as this is a mental process to solve problems, beneficial in a variety of fields like math, engineering, and computer science [13, 15, 25].

Computational Thinking is a problem-solving model that, as the name implies, can be described as a way of thinking. To its development, a variety of concepts need to be mastered, and mental training is required. This type of mind reconversion can be difficult, making its introduction from a young age beneficial. Consequently, this led researchers to believe that Computational Thinking should be taught since kindergarten [15, 26]. Nonetheless, even with children, there are multiple obstacles to overcome. To successfully develop Computational Thinking it is fundamental to adopt suitable evaluation methods, school programs, and learning resources. A learning resource, to be efficacious, should be selected based on the subject and even the student in question.

Games were always attractive to young and even older people, since early years with traditional games to the more common digital ones. Therefore, it makes sense to explore multiple purposes for games than just entertainment. With this, Game-Based Learning appeared and was already proven to be useful in various researches [10, 23]. The increasing popularity of games and multiple characteristics and learning principles embedded in them led us to opt for this method to develop Computational Thinking.

Despite games being already a fun activity, we believe it is possible to boost students' motivation and learning experience from games by choosing the most suitable for each student. As different people have distinct tastes in clothes and food, it should be expected to have distinct preferred game types and learning styles. Therefore, makes sense to analyze what games are best for each individual and to implement that knowledge in the classrooms. For this to be achievable, first needs to be defined a classification for games, in order to have different game types to match with the players. To characterize the different aspects of a game, an ontology is being constructed. Secondly, it is necessary to draw the students' profile having into consideration various aspects like age, gender, country, education level, and personality features. If we connect the profile with game characteristics it should be possible to establish a pattern between players and game types.

Considering that, with age, people tend to have more difficulties in changing the way their minds process information, it is even more crucial to establish the best way to teach Computational Thinking to adults. Therefore, it is essential to analyze the differences between adult and young learning and what are the main challenges that condition adults. Few researches were made to examine the effects of Game-Based Learning in adults [4, 6]. Nonetheless, we believe this method can also be effective on older ages, especially if the person profile is taken into account, as it will evidence the factors that have an impact on adult learning.

This paper after the Introduction is organized in six sections. Section 2 presents research work on Computational Thinking. After that, Section 3 reviews the Game-Based Learning approach and discusses its use for developing Computational Thinking. In Section 4 the ontology, created by this team for the classification of games, is introduced and depicted in the form of a conceptual graph. In Section 5, a central one in our paper, it is discussed how player's profile relate to the enjoyment of games, and the consequent impact on the students learning result. Also in this section, we discuss the way we intend to follow to adequate games to students' profile. Section 6 discusses the factors that need to be considered when applying this approach to adult learners. The paper is concluded in Section 7.

2 Computational Thinking

Computational Thinking (CT) was first defined as a method for solving problems, designing systems, and understanding human behavior, based on fundamental computing concepts [26]. Later on, a new definition was given for CT, being "the thought processes involved in formulating a problem and expressing its solution as transformations to information that an agent can effectively carry out" [25]. With CT solutions are systematically formulated and with enough clarity that we can instruct a machine to execute them [15].

CT is a model of problem-solving, similar to the Scientific Thinking, and comprises a series of processes, approaches, and attitudes [7, 15]. The development of these concepts leads to an improved thinking process and enhanced solutions for a problem.

The processes involved in solving a problem are [7]:

- **Logical Reasoning:** use the existing knowledge to predict the behavior of a system.
- **Algorithm Design:** create a sequence of instructions to solve a problem.
- **Decomposition:** break down complex problems or processes into simpler parts.
- **Pattern Recognition:** identify similarities between problems and apply the same solution to solve them.
- **Abstraction:** remove unnecessary detail, identifying the essential information to solve a problem.
- **Evaluation:** prove that the consider solution is suitable for solving the problem in question.

It is essential to emphasize abstraction, as it is the most important process in CT [26]. With abstraction we can overcome complex problems, as we can use it to define patterns and, on the opposite side, to generalize specific instances. Computing is nothing more than the automation of abstractions, as we first solve a problem using CT, and then implement/automate the solution on a computer.

As for the approaches that characterize CT, there are [7]:

- **Tinkering:** experiment different strategies.
- **Creating:** design with creativity.
- **Debugging:** find and fix errors.
- **Persevering:** be persistent when facing obstacles.
- **Collaborating:** work as a team.

Lastly, when using CT one should assume the following attitudes [15]:

- **Confident:** trust in one's capacity to overcome problems.
- **Communicative:** communicate with others to discuss possible solutions.
- **Flexible:** deal with changes in the course of solving problems and accept open-ended problems.

Technology is present in almost every part of our lives and is necessary in a large variety of professions. Therefore, it is essential that everyone understands the bases of computation, so they can make the most out of it. This brings to the belief of many researchers, which is that everyone can benefit from the development of CT [15, 25].

CT includes a set of skills, attitudes, and approaches that are fundamental, universal, and transferable [15]. Therefore, the development of CT leads to a better understanding of multiple areas like math, engineering, and physics, due to the higher capability of solving problems, expressing solutions, and making abstractions. On a more obvious subject, CT leads to the improvement of computing capacities, which can promote the better performance of Computer Science students. These advantages led to multiple researchers promote that

CT should be taught in K-12, this is, since kindergarten to secondary education [26, 13, 15]. The early introduction of CT comes from the belief that a young mind can be easier modified, making the development of CT more effective. Furthermore, the interdisciplinary of CT is useful at this stage, since in preschool and elementary subjects are taught to the students combined [15].

Being such a mind-changing process, some issues are raised when trying to introduce CT in schools, like what concepts can students thoroughly learn and educators can properly teach, and at what level should the computer be used to do so [26]. Therefore, it is essential to thoroughly analyze details like the learning resources that should be used, and the evaluation methods. For this, it will be analyzed Game-Based Learning as an alternative to develop CT.

3 Game-Based Learning

Nowadays, games are not just for entertaining and a way of spending free time. More and more, researchers and professionals see the advantages of using games for serious purposes. On health and rehabilitation, games are being used to increase physical exercise and as a therapy treatment [6]. In a similar field, games are useful for training the brain, especially on older adults, improving reasoning skills, memory, concentration, alertness, among others [6]. As for business games, these are used to train professionals and can be a good ally for companies [6].

On education, a game can be used as a Learning Resource (LR). LRs are instruments of presentation and transmission of educational subjects. With LR it is possible to put in practice previous knowledge, acquire new knowledge, increase motivation, develop creativity, along with other advantages [3]. These resources can consist of images, maps, diagrams, articles, books, games, posters, among others.

Games are a big part of many students' life, as some spend hours of their free time playing them. This and other factors of students' daily routines, embedded with technology, can lead to a lack of interest in school activities. The traditional learning tools are not stimulating enough to hold their engagement, even becoming outdated [20]. Therefore, Game-Based Learning (GBL) emerged as a strategy to motivate students, being a concept extensively explored by researches [23]. GBL is a technique where games are used as LRs to develop a specific topic, allowing the subject to actively learning. Playing a game leads to learning due to a series of learning principles embedded in games that are activated while a person plays [11, 20]. For accessing a complete list of these principles fully detailed, one should consult the 36 Learning Principles defined on *What Video Games Have to Teach Us About Learning and Literacy* [10]. Some of the most important principles enunciated by multiple researchers are [11, 10, 20]:

1) apply previously learned knowledge, leading the student to develop prior learning;
2) use the feedback given about the student's progress/mistakes, helping him to overcome a difficult problem and to get excited when achieving results in the game;
3) employ the same approach to solve different problems, leading the student to develop problem-solving capabilities;
4) try multiple techniques to solve a problem, learning through practice, failure, reflection, and repetition;
5) interact with other, encouraging unity in a team environment.

These Learning Principles defined in games bring even more sense to the use of GBL to develop CT. For instance, similarities can be found between the items previously described and CT concepts. Items 1 and 3 are related to Pattern Recognition, as previous knowledge is

applied and transferred between problems. Item 4 is associated with Evaluation and Critical Thinking, as it is done experimentation and evaluation of results. It is also worth to mention that, as a game is an immersive activity, it can be more powerful to improve the thinking process. Brain training games are one example of a similar application targeted to develop the mind [6].

Various studies were conducted regarding GBL and its effects on students. Multiple researchers even concluded that this method can be more motivational and more effective for most students to retain knowledge [11, 12, 23]. Nonetheless, GBL should be a complement to the current learning system and never a substitute. For its proper use, it is necessary to ensure students are motivated by the games themselves and not just the competition between peers and that students can obtain knowledge by playing a given game.

Additionally, GBL can also be used as an instrument in adult learning to teach older students, benefiting both learners and facilitators [4, 6]. The majority of the conducted studies have focused on children, teenagers, and young adults [6]. Nonetheless, there has been a growing interest in the potential of GBL on adult education and informal learning, due to the increase of both the life expectancy and the number of adults engaging in learning experiences post K-12, but also the influence of technology on the daily life.

However, with the introduction of technology and games in adult classrooms, some difficulties may arise. For instance, difficulty with acceptance by older users since playing games may also imply having too much "fun" and lack of seriousness in the classroom [1]. In addition, difficulties with interface devices (particularly those with small buttons or writing) that may be challenging for adults with a mobility or visual impairment [6]. Lastly, a considerable period of adaptation since "Adults who have not been familiar with the computer in their youth and who are not scientifically oriented tend to be a little overanxious about using one" [17]. This subject will be further explored in Section 6.

4 Game Classification

Numerous aspects can be considered when defining a game. Among the different categories, some are of easy decision, as others can vary depending on the person classifying the game [5]. On all these aspects, the one that raises more discussion is the game genre. Various researchers and game designers came up with different possibilities of game genres, being complicated to reach a consensus. For example, game designer Tracy Fullerton [9] divided games into action, strategy, role-playing, sports, racing/driving, simulation/building, flight and other simulations, adventure, educational, children's, casual, and experimental. On a research to establish the connection between genres and usability of video games it was used a set of 6 genres to categorize games, being them [22]: role-playing games (RPGs), sports/racing, first-person shooter/tactical shooter, action, strategy, and adventure. Although many variations, a set of genres are very common in different classifications, like action and strategy.

To develop a proper classification for games, we believe the best method is to construct an ontology. Vastly used by researchers and companies, ontologies are an efficient approach to describe a domain. Therefore, *OntoJogo* (Figure 1) is an Ontology that is being created for Game Classification based on already existing ontologies for games like the ones presented on references [28, 5, 16].

At this point, the ontology divides games into digital and non-digital. Both of these types have some features in common. Therefore, a *game* can be defined by *player number*, *genre*, *story progression*, and *game mode*. The *player number* is presented as the accepted

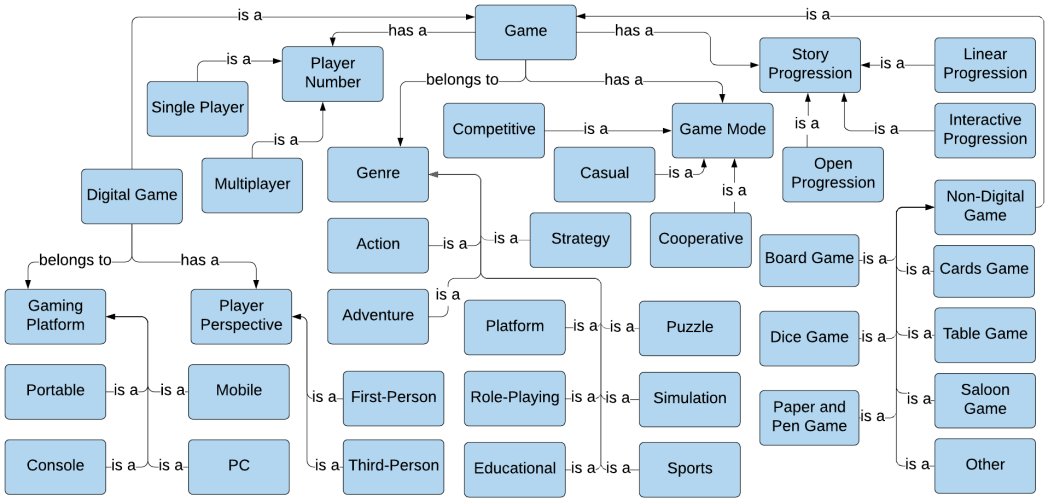


Figure 1 *OntoJogo*: Ontology for Game Classification.

way of *single-player* or *multiplayer*. As for *game genres*, we opted to divide them into *action*, *adventure*, *educational*, *platform*, *puzzle*, *role-playing*, *simulation*, *sports*, and *strategy*. As said before, genre is not an easy category to establish, being this the set that makes more sense for the work being done. The *story progression* refers to the level of exploration the player can do in the game and how the player's actions can impact the narrative. In a *linear* game, the player is fully oriented throughout the narrative, for example, by crossing levels. Therefore, the end and progression is equal for every player. When playing an *interactive* game, the end of the game can be different depending on the player's choices. The player still has a guidance but is always presented with possibilities to pick. On the other extreme, in a *open* game, the player can do what he feels like doing, not having any orientation and facing infinite possibilities. Finally, the *game mode* indicates the type of objectives the player intends to achieve. Depending on the *game mode*, the player posture should differ. In a *casual mode*, there is no competition, and the player can have a more relaxed attitude. In a *competitive mode*, the player is facing adversaries, being expected for him to try and win the game. In a *cooperative mode*, the player should make his best have a team spirit and be social.

Digital games is one that implies the use of technology. This type of game has as features the *gaming platform* and *player perspective*. The *gaming platform* can be *console*, *PC*, *mobile*, and *portable*. As for the *player perspective*, it is related to the player's perception of his character. In a *first-person* perspective, the player can not see his character and has a perception of the environment similar to the real world. In a *third-person* perspective, the player observes his character, creating an extra mental distance between him and the game. *Non-digital game* is one that does not require any technology, being more traditional. At this point, we divide this type of game into *board game*, *dice game*, *cards game*, *table game*, *paper and pen game*, *saloon game*, and *other*. *Saloon game* is the type that does not require any artifact, like a game that only involves singing or talking. The concept *other* was included for non-digital games not considered on the ontology, as a complete definition of this category could be very extensive.

These were the characteristics we believe to be more impacting on the player experience. Additionally, it is relevant to have in mind that most of the categories' options are not exclusive. For example, a game can be cooperative and competitive at the same time if players are part of teams competing against each others.

5 Games and Players Profile

As stated before, to use games as educational tools, it is essential to guarantee students are motivated by the games they play. For this, it is necessary to understand what are the factors that motivate each student and how to hold that motivation through time.

Examining first the part of maintaining the student motivated, it is interesting to analyze the *flow* concept. *Flow* is the optimal experience, where the subject lies between boredom and anxiety, not leading to either of the states [8]. When a player reaches this state becomes fully absorbed in the game. To achieve the state of *flow*, some characteristics are essential for the game. For example, the game must have clear goals and provide enough feedback [8]. Additionally, the level of customization in the game, although not required, can also have an impact. The more a player can manipulate characters, the more he constructs an identity, feeling motivated to keep playing for longer periods [11].

Aside from keeping a student motivated, it needs to be analyzed why that motivation appears in the first place. The most usual method to aboard this problem is by introducing questionnaires. Different approaches can be applied with questionnaires: open response surveys on what motivates the player [14]; closed response surveys about the player's motivation in a specific game [18, 27]; observation of the player's behavior while playing a game [24]. All these methods have the purpose of making a connection between the players' profile and types of games that motivates them. This connection is significant not just for the motivation factor, but to understand if playing a particular type of games can have a positive or negative effect on a person. An example of this is the revolt towards violent games. Some suggest that instead of concluding violent games turn the players more violent, it should be analyzed if the players that choose them already tend towards violence. This means that instead of negatively labeling a game, we should try to conclude if the game is suited for all players [18].

A traditional approach to analyze players is by establishing player types. One of the most popular and used categorization divides players into four different classes [2]: achievers, explorers, socializers, and killers. Although very acknowledging, some consider it not the best system to use in research, as it was never proven the player types to be independent. Furthermore, as player types are often defined with one personality trait on focus, it could lead to bias results [18, 27]. On a different strategy, one can analyze several aspects of the person, including age, gender, and personality features. Some researches were done to examine gender and age as influencing factors on players' motivation. For example, on Yee [27] work, conclusions were made on gender differences on social and achievement components. To examine the personality of the student, a possible approach is the Five-Factor Model of personality, which consists of analyzing the following traits [24]: openness to new experience, conscientiousness, extraversion, agreeableness, and neuroticism.

Although some work was already done on the subject and some conclusions were made, there is a need for future research to establish a connection between players' characteristics and the enjoyment of playing games [24]. For this, more features of the player should be considered, namely: sociodemographic factors like age, gender, nationality, level of education, family members, among others; personality factors like persistence, socialization, positivity, among others; and previous experiences with playing games.

To relate games with players, we intend to use the presented ontology, *OntoJogo*, together with a questionnaire for profiling players. A web platform will be developed to take advantage of these resources. Games and students should be registered in this platform, and the reaction of each student to a particular game should be recorded. With the accumulation of results, and resorting to pattern recognition and machine-learning algorithms, it should be possible to determine what are the most suitable games for each student.

6 The Bridge Between Young and Adult Learning

As mentioned previously in Section 3, GBL can also be used as an advantageous instrument in adult learning and education, possessing however its own set of difficulties over K-12 regular school. Therefore, when looking to expand GBL to older learners one has to consider the particularities of adult learning and its differences over young learning in order to successfully adapt the extensive research, programs and materials already available to serve older audiences.

Foremost, when talking about adult learning and education it is important to distinguish said concepts. Adult Education refers to the teaching of adults, that is, instructing/giving lessons to adult students with the involvement of a teacher and according to a curricular program, guide or plan of education. On the other hand, Adult Learning refers to the continuous process of learning and developing skills and knowledge throughout an adult's life.

Another related concept of extreme relevance when it comes to adapt GBL to adults is literacy. According to the OECD's (Organisation for Economic Co-operation and Development) PIAAC study (Programme for the International Assessment of Adult Competencies) literacy is defined as *"the ability to understand and use information from written texts in a variety of contexts to achieve goals and develop knowledge and potential"* [21]. It is important to take into consideration the student's level of literacy in order to be able to identify and apply the games more suited to its skills, since literacy among adults varies from an extensive range. The PIAAC study defined 5 proficiency levels that can be use as guide to the categorization of games according to this parameter. The levels are as follows [21]:

- **Level 1:** Lowest level of literacy. People at this level must be able to recognise basic vocabulary, determine the meaning of sentences and read short texts.
- **Level 2:** Ability to make matches between the text and information, paraphrasing or low-level inference.
- **Level 3:** Knowledge and skill in interpreting and constructing meaning across dense or lengthy texts; identifying, interpreting and evaluating pieces of information at various levels of inference.
- **Level 4:** People who display ability to integrate, interpret, synthesise, infer from complex or lengthy texts; apply background knowledge and identify and understand non-central ideas in texts.
- **Level 5:** Maximum level of literacy. Knowledge and skill in searching for, integrating, synthesising and selecting key information across multiple dense texts; making high-level inferences.

Lastly, regarding the differences between young and adult learning, there are two main models of learning assumptions: Pedagogy and Andragogy. Pedagogy (from the Greek meaning *"child leading"*) is defined as the art and science of teaching children and Andragogy, in opposition, is defined as the art and science of teaching adults [19]. Knowles [19] compares Pedagogy's and Andragogy's assumptions regarding the concept of the learner, the role of the learners' experience, readiness to learn and orientation to learning, allowing for a better understanding of both models. The comparison is as follows:

- **Concept of the learner:** In Pedagogy the learner has a dependent role whereas in Andragogy the learner matures from dependency to self directedness.
- **Role of the learners' experience:** In pedagogy the learners' personal and life experience is of reduced importance. In Andragogy, the learner's experience is valued, adults attach more meaning to learn they gain from experience.

- **Readiness to learn:** In Pedagogy students are willing to learn what is decided they should, and most are capable of the same learning that of their peers whereas in Andragogy students are willing to learn that which will help them on their real-life problems.
- **Orientation to learning:** When it comes to Pedagogy, learners are subject-centered in their orientation to learn. On the other hand, in Andragogy, learners are performance centered and see education as a process of developing skills to better their life.

In addition to the concepts stated above from an educational perspective, it is also essential to take into account the student's profile. Since adults have a more structured personality than young students, considering their profile when applying these new teaching/learning methods could be decisive to their effectiveness.

7 Conclusion

The difficulties found in general by young people and adults to solve problems manually or by computer could be overcome by training Computational Thinking. Since this is a mind process, the earlier this training occurs the easier it is to have a positive effect on problem solving activities.

Guided by this belief, we started a research project aimed at finding ways to realize the referred approach. First of all, we discovered that the choice of accurate learning resources is a crucial step. Because Game-Based Learning is an immersive activity, we considered it to be an effective way to motivate students and change minds. That feeling proved to be true after a careful literature review. However, games used as learning resources must be carefully chosen to fit properly in the students' profile. For this effect, we proposed an ontology to classify games and a series of features to have in consideration when analyzing the students' profile. In addition, it is also necessary to determine how Computational Thinking can be developed in older ages. In this paper, it was given a perspective on how Game-Based Learning can be employed in adults. Furthermore, it was analyzed what differs between young and adult learners, and how the use of proper resources for each student can also be beneficial to deal with this matter.

As future work we intend to develop the questionnaires to identify the students' profile and to determine the appropriate similarity algorithms capable of finding the connection between them and games.

References

1. Bryce O. Anderson, Michelle N. Anderson, and Thomas A. Taylor. New territories in adult education: Game-based learning for adult learners. *Adult Education Research Conference*, 2009.
2. Richard Bartle. Hearts, clubs, diamonds, spades: Players who suit muds. *Journal of MUD research*, 1(1):19, 1996.
3. Rona Bušljeta. Effective use of teaching and learning resources. *Czech-Polish Historical and Pedagogical Journal*, 5(2):55, 2013.
4. Rosemary S Caffarella. Planning programs for adult learners, 2002.
5. Yuri Gomes Cardenas et al. *Modelo de ontologia para representação de jogos digitais de disseminação do conhecimento*. PhD thesis, Universidade Federal de Santa Catarina, 2014.
6. Nathalie Charlier, Michela Ott, Bernd Remmele, and Nicola Whitton. Not just for children: game-based learning for older adults. In *6th European Conference on Games Based Learning, Cork, Ireland*, pages 102–108, 2012.
7. Council for The Curriculum Examinations and Assessment. *Computing at school: Northern Ireland curriculum guide for post primary schools*. Computing at School, 2018.

8 Mihaly Csikszentmihalyi. Flow and the psychology of discovery and invention. *HarperPerennial, New York*, 39, 1997.

9 Tracy Fullerton. *Game design workshop: a playcentric approach to creating innovative games*, chapter 15, pages 474–481. AK Peters/CRC Press, 2018.

10 James Paul Gee. *What Video Games Have to Teach Us about Learning and Literacy.* Palgrave Macmillan, 2003. doi:10.1108/et.2004.00446dae.002.

11 James Paul Gee. What video games have to teach us about learning and literacy. *Comput. Entertain.*, 1(1):20–20, October 2003. doi:10.1145/950566.950595.

12 James Paul Gee. Surmise the possibilities: portal to a game-based theory of learning for the 21st Century. *Clash of realities*, page 33, 2008.

13 Shuchi Grover and Roy Pea. Computational thinking in K–12: A review of the state of the field. *Educational researcher*, 42(1):38–43, 2013.

14 Karla R Hamlen. Children's choices and strategies in video games. *Computers in Human Behavior*, 27(1):532–539, 2011.

15 E Hunsaker. No Title. In Ottenbreit-Leftwich and R Kimmons, editors, *The K-12 Educational Technology Handbook*, chapter Computatio. EdTech Books, 2018. URL: https://edtechbooks.org/k12handbook/computational_thinking.

16 Fabricio Janssen, Renata Araujo, and Fernanda Baião. Uma proposta de Ontologia de gêneros e narrativas em jogos digitais para a Game Ontology Project (GOP). *RelaTe-DIA*, 12, 2019.

17 Peter Jarvis. *Adult and continuing education: Theory and practice.* Psychology Press, 1995.

18 Daniel Johnson and John Gardner. Personality, motivation and video games. *ACM International Conference Proceeding Series*, pages 276–279, 2010. doi:10.1145/1952222.1952281.

19 Malcolm S Knowles et al. The modern practice of adult education: From pedagogy to andragogy (revised and updated). *Englewood Cliffs, NJ: Cambridge Adult Education*, 1980.

20 Diana Oblinger. The next generation of educational engagement. *Journal of interactive media in education*, 2004(1), 2004.

21 OECD. *The survey of adult skills : reader's companion.* OECD Publishing, Paris, 2019.

22 David Pinelle, Nelson Wong, and Tadeusz Stach. Using genres to customize usability evaluations of video games. In *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, Future Play '08, pages 129–136, New York, NY, USA, 2008. ACM. doi:10.1145/1496984.1497006.

23 Han-Yu Sung and Gwo-Jen Hwang. A collaborative game-based learning approach to improving students' learning performance in science courses. *Computers & education*, 63:43–51, 2013.

24 Giel Van Lankveld, Pieter Spronck, Jaap Van Den Herik, and Arnoud Arntz. Games as personality profiling tools. *2011 IEEE Conference on Computational Intelligence and Games, CIG 2011*, pages 197–202, 2011. doi:10.1109/CIG.2011.6032007.

25 Jeanette Wing. Research notebook: Computational thinking—What and why. *The Link Magazine*, pages 20–23, 2011.

26 Jeannette M Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.

27 Nick Yee. Motivations for play in online games. *CyberPsychology & behavior*, 9(6):772–775, 2006.

28 José P Zagal, Michael Mateas, Clara Fernández-Vara, Brian Hochhalter, and Nolan Lichti. Towards an ontological language for game analysis. *Worlds in play: International perspectives on digital games research*, 21:21, 2007.

Using Property-Based Testing to Generate Feedback for C Programming Exercises

Pedro Vasconcelos
Computer Science Department, Faculty of Science, University of Porto, Portugal
LIACC, Porto, Portugal
pbv@dcc.fc.up.pt

Rita P. Ribeiro
Computer Science Department, Faculty of Science, University of Porto, Portugal
LIAAD-INESC TEC, Porto, Portugal
rpribeiro@dcc.fc.up.pt

Abstract

This paper reports on the use of property-based testing for providing feedback to C programming exercises. Test cases are generated automatically from properties specified in a test script; this not only makes it possible to conduct many tests (thus potentially find more mistakes), but also allows simplifying failed tests cases automatically.

We present some experimental validation gathered for an introductory C programming course during the fall semester of 2018 that show significant positive correlations between getting feedback during the semester and the student's results in the final exam. We also discuss some limitations regarding feedback for undefined behaviors in the C language.

2012 ACM Subject Classification Social and professional topics → Student assessment; Software and its engineering → Software testing and debugging; Software and its engineering → Domain specific languages

Keywords and phrases property-based testing, C language, Haskell language, teaching programming

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.28

1 Introduction

This paper reports on the use of property-based testing for automatically generating feedback to exercises in an introductory C programming course. As part of weekly worksheets, students were given assignments with automatic tests; they submitted solutions through a web-system and got feedback automatically. The tests assessed functional correctness, reporting the test data leading to a failure in case of wrong answers. Students could also perform multiple attempts, either to fix mistakes or to try alternative solutions.

Instead of fixed test suites, test cases are generated randomly from *properties* in a test script. This not only makes it easier to conduct more tests (thus potentially find more mistakes), but also allows reporting failed test cases: because new tests are generated, students cannot "cheat" simply by submitting solutions that are over-fitted to previously-reported cases. Furthermore, *shrinking* heuristics can be used to simplify failing cases automatically; this is helpful because randomly generated data often contains "noise" that is not relevant to the failure. Reporting shorter examples should help students debug their code and clarify misunderstandings.

This paper is structured as follows: Section 2 presents some related work; Sections 3 and 4 review property-based testing and describe our testing framework; Section 5 presents the experimental results; and Section 6 presents conclusions and directions for further work.

2 Related work

There has been increasing interest in the use of automated testing as an aid for teaching programming. Keuning et al. present a systematic comparison of tools for automatic feedback generation for programming exercises [10].

Gao et al. employ *concolic testing* to generate automatic feedback for a C programming course [8]; this approach is based on symbolic execution, and when compared to the one presented here, requires extra work to setup and process each exercise.

Fisher and Johnson describe the use of formal specifications as test case generators for Java programs [7]; this was done in the context of a software engineering course where understanding and writing specifications are part of the learning outcomes.

Our work is more closely-related to the one by Earle et al. [4] on using property-based testing to automatically assess Java programs. The main differences are: 1) we use Haskell instead of Scala for specifications; 2) we test C rather than Java programs and deal with the C specific issues such as undefined behaviours; and 3) we focus on functional correctness rather than grading.

3 Property-based testing

Property-based testing consists of specifying general assertions about software units and using these as test oracles; the actual test data can then be automatically generated (e.g. sampled randomly). The first property-based testing tool was the *QuickCheck* library for the Haskell language [5]. Similar libraries have since been developed for other languages, e.g. Erlang [1], Scala [12] and Python [11].

Properties are universally quantified assertions; testing these requires generating values and checking the assertion for each case. Passing a suitable large number of tests increases confidence in the correctness of the property. A single failed test, however, provides evidence that the property does not hold. Moreover, the failed test case can then be simplified automatically by applying "shrinking" heuristics for generated values.[1] Shrinking is useful because it can automatically remove irrelevant details from randomly generated data, leading to shorter and more insightful counter-examples.

Libraries for property-based testing provide functions for defining properties and generators from simpler ones and to control the number of tests and size data, making it possible to apply this methodology to real programs [3, 9].

4 Testing framework

4.1 Overview

Exercises consisted of individual C functions rather than complete programs. Alongside the problem description, the instructor sets up a test script written using *codex-quickcheck*[2], a custom version of the Haskell QuickCheck library with functionality to ease generating C

[1] For example, a strategy for shrinking strings is to try substrings of smaller length, or to replace some characters with strictly smaller ones.

[2] Available at `https://github.com/pbv/codex-quickcheck`.

types, testing C functions and producing student-friendly reports. This library is part of the *Codex* system for programming exercises[3] but can be used independently. The testing workflow consists of:

1. compiling the student code to an object file;
2. compiling and linking the test script together with the object file;
3. running the executable (under a sandbox) to produce an accepted/failure report.

Submissions are classified as follows:

CompileError rejected attempt due to a compiler error or warning;

RuntimeError, TimeLimitExceeded or MemoryLimitExceeded the execution was aborted due to a runtime error or resource exhaustion;

WrongAnswer testing found a witness that falsifies a property;

Accepted all tests passed.

Because the exercises were not computationally intensive, the turn-around for student feedback was quick (typically 2-4 seconds). Since students could edit code directly on the web interface and there was no penalty for repeated submissions, many submissions were made even for short assignments.

4.2 Example: testing the median function

Listing 1 Test script for the median function exercise.

```
1  import Data.List (sort)
2
3  foreign import ccall "median" median :: CInt -> CInt -> CInt -> CInt
4
5  prop_correct
6    = testing "median" $
7      forArbitrary "a" $ \a ->
8      forArbitrary "b" $ \b ->
9      forArbitrary "c" $ \c ->
10         median a b c ?== sort [a,b,c] !! 1
11
12 main = quickCheckMain prop_correct
```

Listing 1 presents the script for testing a sample exercise: *write a function to compute the median of three numbers*. Line 3 imports the C function to be tested using the standard Haskell foreign-function interface. Lines 5–10 specify the correctness property: the median of a, b, c is the value at index position 1 (i.e. the middle element) of the sorted list of values.[4] The `forArbitrary` function introduces a quantified property using the default generator (for C integers, in this case) and also names the variable for reporting. The assertion operator `?==` compares the left-hand side with the (expected) right-hand side, producing a suitable message otherwise.

Listing 2 shows a hypothetical feedback for a submission that gives a wrong result when two arguments are equal. Reports always follow this format: the name of the function being tested, the names and values of arguments and the expected and obtained results. Students can use reports to clarify their understanding of the exercise, or as a starting point for debugging, without any knowledge of Haskell or property-based testing.

[3] Available at `https://github.com/pbv/codex`.
[4] `!!` is the list indexing operator in Haskell.

Listing 2 Example feedback for the median function exercise.

```
*** Failed! Falsified (after 1 test and 2 shrinks):
Testing median with:
        a = 0
        b = 0
        c = 1
Expected:
        0
Got:
        1
```

In principle, properties are quantified over all possible values; in practice, they are tested with a finite sample of randomly-generated test cases (100 by default). The number of tests and the maximum size of values are configured as metadata for exercises; it is also possible to fix the seed for pseudo random-number generation (and thus get reproducible results). Test cases are generated with increasing sizes, so that the smaller examples are tried first. Furthermore, shrinking simplifies failed examples before reporting.

4.3 Custom generators and shrinking

The previous example used the default generators for integers. In general, we need to be able to define properties using custom generators. Listing 3 presents a test script for such an example, namely, a function that checks if a string is a "strong password" according to the following rules: it should have at least 6 characters and contain one lowercase, one uppercase and one digit character.

Lines 6–8 define a functional wrapper for the C function to be tested; it uses `withCString` to convert a Haskell string into a C character buffer[5], ensuring proper deallocation. Lines 10–13 define a Haskell specification for the solution; lines 15–18 specify the correctness property using `forAllShrink` instead of `forArbitrary`; this allows using a *custom generator* (line 20) and *shrinking function* (line 21) that generates strings containing only a subset of printable ASCII characters. The generator is defined using functions `listOf` and `choose` from the QuickCheck library; we use `shrinkMap` to apply the default shrinking for strings filtering out characters outside the desired range.

Listing 3 Test script for the strong password exercise.

```
1  import Data.Char(isUpper, isLower, isDigit)
2
3  foreign import ccall "strong_passwd"
4    strong_passwd :: CString -> IO CInt
5
6  strong_passwd_wrapper :: String -> CInt
7  strong_passwd_wrapper str
8    = runC $ withCString str strong_passwd
9
10 strong_spec :: String -> Bool
11 strong_spec str
12   = length str>=6 && any isUpper str &&
13     any isLower str && any isDigit str
14
```

[5] Haskell strings are linked-lists rather than contiguous buffers.

```
prop_correct
  = testing "strong_passwd" $
    forAllShrink "str" genPasswd shrinkPasswd $ \str ->
       strong_passwd_wrapper str ?== fromBool (strong_spec str)

genPasswd = listOf (choose ('0', 'z'))
shrinkPasswd = shrinkMap (filter (\c -> c>='0' && c<='z')) id

main = quickCheckMain prop_correct
```

Listing 4 presents reports based on a real student attempt that incorrectly assumed that all characters in the string must be letters or digits. The top report illustrates a random test case obtained (before shrinking), while the bottom one is obtained after shrinking; we report the later to students. This example illustrates the effectiveness of shrinking to automatically produce more insightful counter-examples.[6]

Listing 4 Example feedback for the strong password function exercise with shrinking disabled and enabled.

```
*** Failed! Falsified (after 16 tests):
Testing strong_passwd with:
        str = "gSvF<NiXz]BH_"
Expected:
        0
Got:
        1

*** Failed! Falsified (after 16 tests and 7 shrinks):
Testing strong_passwd with:
        str = "aaaaA_"
Expected:
        0
Got:
        1
```

4.4 Mitigating undefined behavior

One of the challenges of teaching the C language is the need to alert students to avoid inadvertently causing *undefined behaviors* (UB). It is important to catch UB when doing automated testing because they can lead to puzzling results (e.g. non-deterministic answers across operating systems, compiler versions, etc.). While ensuring the complete absence of UB is quite difficult, we employed some simple mitigation techniques using the GNU C Compiler:

- *enabling exhaustive warnings* and treating them as compile-time errors;
- *enabling optimizations* (e.g. `-O1`) also enables static checks about potential errors (e.g. uses of uninitialized variables);
- *enabling the UB sanitizer* (e.g. `-fsanitize=undefined`) for introducing runtime checks for some UB (e.g. division by zero and integer overflows).

[6] The string found is, in fact, a *local minimum* that distinguishes the student's attempt from a correct solution: it has length 6, at least 1 lower and uppercase letter, but no digit.

Another mitigation technique is to perform all memory management and array initialization from the Haskell side (e.g. the `withCString` function). The *codex-quickcheck* library provides a *checked* initialization function that places random "canaries" [6] outside the array boundaries and checks for overwrites. This allows detecting and reporting off-by-one index errors in student code that performs array writes; however, it is not so effective at catching index errors for array reads (see also the discussion in Section 5.2 regarding exercise ex8_2).[7]

4.5 Developing specifications

We developed test scripts for 15 exercises covering elementary programming concepts (e.g. function definitions, conditionals, loops, arrays and strings). Most test scripts are short (e.g. under 50 lines) and required between 15 to 60 minutes to develop. Moreover, properties and generators developed for one exercise can often be easily adapted to related ones. The largest script has 116 lines (exercise ex9_5 in Table 1); this is due to code for generators of matrices that are "magic squares" and ones that fail each of the necessary conditions.

In order to check adequate test case distribution, first we developed the correctness properties alone, and subsequently collect statistics to adjust the testing parameters and generators. For the example of Section 4.3, we can count the percentage of "small" test cases (i.e. length less than 6) by introducing into line 18 of Listing 3:

```
classify (length str < 6) "small" $ ...
```

To also count which combinations of conditions are tested, we further add:

```
collect (any isUpper str, any isLower str, any isDigit str) $ ...
```

Running the test script with a reference solution we obtain:

```
+++ OK, passed 100 tests (20% small)
81% (True,True,True)
 9% (True,True,False)
 5% (False,False,False)
 2% (False,True,False)
 1% (False,False,True)
 1% (False,True,True)
 1% (True,False,False)
```

This shows that 20% of the generated strings were small and 81% satisfied all conditions; also, some combinations of conditions were poorly tested and one combination was untested. We can now improve this distribution by changing the test data generation; for this example it suffices to reduce the maximum data size (i.e. the maximum string length) and increase the number of tests.

5 Experimental evaluation

Throughout the semester students could submit exercise solutions using the web system; they were not penalized for multiple attempts and could also continue submitting even after having an accepted solution (e.g. to experiment with alternatives). Submissions were used as formative rather than summative assessment; however, a minimal number of correct submissions was required to qualify for the final exam.

[7] The GCC address sanitizer (`-fsanitize=address`) cannot help here because it does not track heap overflows in memory managed by the Haskell runtime system.

Table 1 Summary of exercises, total number of attempts, percentage of wrong answers and median (maximum) number of attempts per student.

Exercise	Description	#Attempts	%WrongAns	#Attempts per student
ex3_8	median of 3 integers	1024	17.2%	5.0 (41)
ex3_9	compute integer powers	1288	28.3%	3.5 (84)
ex4_5	sum integer divisors	740	30.4%	3.0 (28)
ex4_7	determine next leap year	1048	23.5%	4.0 (48)
ex5_8	approximate a power series	1103	45.6%	4.0 (67)
ex6_4	initialize an array	484	14.0%	2.0 (36)
ex6_8	test repeated values in array	712	38.2%	2.0 (33)
ex7_5	check strong passwords	767	27.8%	3.5 (24)
ex7_10	filter positive values in array	656	42.5%	1.5 (54)
ex8_2	check if an array is ordered	830	50.1%	3.0 (62)
ex8_7	insertion into an ordered array	1192	40.4%	4.0 (63)
ex9_2	check if 2 strings are anagrams	848	32.1%	3.0 (33)
ex9_5	check magic squares	785	47.3%	2.0 (74)
ex10_4	array max and min using pointers	539	42.3%	2.0 (25)
ex10_9	find character in a string using pointers	368	12.0%	1.0 (21)

Ideally, we would assess the effect of automatic feedback by comparing the learning results of students in two groups, one group getting the automatic feedback while the control group getting no feedback. However, this would be undesirable due to the differentiate treatment of students in the class. Instead, we followed an approach similar to Ahadi et al. [2] and performed a correlation analysis between students' submissions and their final results. In particular, we measured correlations between the *total number of attempts* and the *number of wrong answers* on each exercise and the students' final exam scores.

5.1 Experimental setup

The exam was attended by 152 students which, during the semester, had performed a total of 12384 submissions for the 15 proposed exercises. From these submissions, we gathered, for each student and each exercise: the number of attempts made (#Attempts); and the number of attempts classified with *Wrong Answer* (#WrongAns).

The data set consists of 2280 (152×15) observations. Table 1 lists, for each exercise, the total number of submissions made, the percentage of *Wrong Answers*, the median and the maximum number of attempts per student. Observe that some exercises have quite more attempts than others. Moreover, the percentage of *Wrong Answer* varies considerably. Note also that the median number of attempts per student is much smaller than the maximum, which indicates that only a few students perform such large amount of attempts.

For each of the 15 exercises, we classified students according to the following scenarios:

1. the student made $< 2^n$ attempts;
2. the student made $\geq 2^n$ attempts;
3. the student made $< 2^n$ *Wrong Answer* attempts;
4. the student made $\geq 2^n$ *Wrong Answer* attempts.

The maximum number of attempts per student for any exercise was 84 (cf. Table 1); thus, we have considered n ranging from 0 to 6 in the above scenarios.

We performed a *Pearson correlation test* for each of the binary variables generated in the 4 scenarios above and the binary variable indicating whether the student scored above the overall median grade of the exam. As all the variables involved in the correlation analysis are binary, this corresponds to obtaining the *phi* correlation coefficient. This coefficient is obtained on the basis of a contingency table for two binary variables, as exemplified in the Table 2. This table contains the number of students satisfying each of the four possible combinations for the two binary variables under analysis. One of the variables refers to one of the 4 scenarios for a given exercise and the other refers to the obtained score w.r.t. to the overall median score of the final exam. This means that each of the 15 exercises was tested against the final exam score, in all the possible scenarios.

Table 2 Contingency table for exercise X in the context of scenario Y w.r.t the overall final exam score median.

		final exam score $\geq$ median?	
		yes	no
exercise X meets criterion of scenario Y?	yes	a	b
	no	c	d

Based on Table 2, the *phi* coefficient is obtained by the following formula:

$$phi = \frac{ad - bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}} \tag{1}$$

The correlation tests were performed using the function `cor` from the R statistical language [13]. The following criteria were used for pruning the contingency tables (similar to the ones used in [2]):

1. to avoid over-fitting [14], contingency tables with any cell value less than 5 were pruned;
2. contingency tables with negative *phi* were pruned; for each of these there is a "mirror image" table for the reversed criteria with a positive *phi* value;
3. if two contingency tables have *phi* values that differ less than 0.01 and one is more general than the other, the less general one was pruned (e.g. "$\geq$ 2 attempts" is more general than "$\geq$ 4 attempts");
4. if two contingency tables have *phi* values that differ more than 0.01 and the more general one has a higher *phi* value, the less general one was pruned.

5.2 Main results and discussion

Table 3 presents the results obtained by our tests that are significant with a 95% confidence level, i.e. with a p-value < 0.05.

One might expect that solving exercises in fewer attempts would correlate with better final results, but we detected more correlations with "$\geq$" criteria; this was also observed in the previous work [2] and we conjecture that similar explanations apply here:

- some exercises with a high number of total attempts also exhibit high correlation with final results (e.g. ex5_8, ex8_7); perhaps these exercises trigger difficulties that students need to experience in order to improve their understanding;
- students could submit attempts out of class, thus if they performed too few submissions it could simply indicate they were getting solutions from someone else;
- finally, a high number of attempts could also be an indicator that the student used the system specifically as a study aid for the exam.

Table 3 Significant correlations between the number of attempts or wrong answers by exercise and the final exam score (p-value < 0.05).

Exercise	#Attempts	#WrongAns	*phi*	p-value
ex3_9	*	≥ 1	0.238	0.003
	≥ 2	*	0.190	0.019
ex4_5	*	≥ 2	0.179	0.027
ex5_8	*	≥ 1	0.277	0.001
	≥ 4	*	0.198	0.015
ex6_8	≥ 2	*	0.199	0.014
ex7_5	≥ 1	*	0.229	0.005
ex7_10	*	≥ 1	0.302	0.000
	≥ 1	*	0.272	0.001
ex8_2	*	< 4	0.196	0.016
	< 8	*	0.199	0.014
ex8_7	*	≥ 4	0.239	0.003
	≥ 1	*	0.287	0.000
ex9_2	*	≥ 1	0.267	0.001
	≥ 1	*	0.272	0.001
ex9_5	*	≥ 2	0.303	0.000
	≥ 2	*	0.254	0.002
ex10_4	*	≥ 1	0.174	0.032
	≥ 1	*	0.230	0.004
ex10_9	≥ 1	*	0.200	0.013

The results for exercise ex8_2 show an unusual pattern: criteria with "$<$" conditions correlated better with final results, while criteria with "$\geq$" do *not* exhibit significant correlation. Also, this exercise had an unusually high percentage of *Wrong Answer* outcomes (cf. Table 1). Analyzing the submissions, we observed that a large number of *Wrong Answers* were actually caused by reading past the end of the array. The testing library does not detect this error as an undefined behavior (see Section 4.4) and the test case reported might not be easily reproducible; this might explain why the feedback obtained was less helpful.

For the remaining exercises in Table 3 having a higher number of attempts correlated positively with better student's results, and in particular, for exercises ex3_9 ex4_5, ex5_8, ex7_10 and ex9_5, the number of *Wrong Answer* attempts had a higher and more significant correlation than the number of attempts alone. While these results do not prove causality, they are consistent with the hypothesis that automatically generating feedback has improved students' results.

6 Conclusion and further work

We have presented the use of property-based testing for providing automatic feedback to C programming exercises. Our approach is to use a library in a high-level language for defining properties to generate many random test cases. Furthermore, the test cases can be automatically simplified before reporting to students. Experimental results indicate that there were significant positive correlations between the use of this testing system and the students' final results.

One direction for further work is to integrate an interactive simulator into the feedback loop. Automatically invoking a web-based system such as the *C Tutor*[8] should help students investigate failures (we recommended students performed this step manually in classes). The visualizer could also help students because it explicitly highlights undefined behaviors as errors.

To test interfaces with multiple functions rather than single ones we could use the finite-state machine approach of the Erlang QuickCheck [9]; this could be useful for more advanced C programming courses (e.g. on data structures or operating systems).

References

1 QuviQ AB. Erlang QuickCheck. `http://www.quviq.com/products/erlang-quickcheck/`, 2018. [Online; accessed April 2020].

2 Alireza Ahadi, Raymond Lister, and Arto Vihavainen. On the number of attempts students made on some online programming exercises during semester and their subsequent performance on final exam questions. In *Proc. of the 2016 ACM Conf. on Innovation and Technology in Computer Science Education*, ITiCSE '16, pages 218–223. ACM, 2016.

3 Thomas Arts, John Hughes, Joakim Johansson, and Ulf Wiger. Testing telecoms software with Quviq QuickCheck. In *Proc. of the 2006 ACM SIGPLAN Workshop on Erlang*, ERLANG '06, pages 2–10. ACM, 2006.

4 Clara Benac Earle, Lars-Åke Fredlund, and John Hughes. Automatic grading of programming exercises using property-based testing. In *Proc. of the 2016 ACM Conf. on Innovation and Technology in Computer Science Education*, ITiCSE '16, pages 47–52. ACM, 2016.

5 Koen Claessen and John Hughes. Quickcheck: A lightweight tool for random testing of haskell programs. In *Proc. of the Fifth ACM SIGPLAN International Conf. on Functional Programming*, ICFP '00, pages 268–279. ACM, 2000.

6 Crispin Cowan, Perry Wagle, Calton Pu, Steve Beattie, and Jonathan Walpole. Buffer overflows: attacks and defenses for the vulnerability of the decade. *Foundations of Intrusion Tolerant Systems*, pages 227–237, 2003.

7 Gene Fisher and Corrigan Johnson. Making formal methods more relevant to software engineering students via automated test generation. In *Proc. of the 2016 ACM Conf. on Innovation and Technology in Computer Science Education*, ITiCSE '16, pages 224–229. ACM, 2016.

8 Jianxiong Gao, Bei Pang, and Steven S. Lumetta. Automated feedback framework for introductory programming courses. In *Proc. of the 2016 ACM Conf. on Innovation and Technology in Computer Science Education*, ITiCSE '16, pages 53–58. ACM, 2016.

9 John Hughes. *Experiences with QuickCheck: Testing the Hard Stuff and Staying Sane*, pages 169–186. Springer International Publishing, 2016.

10 Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. Towards a systematic review of automated feedback generation for programming exercises. In *Proc. of the 2016 ACM Conf. on Innovation and Technology in Computer Science Education*, ITiCSE '16, pages 41–46. ACM, 2016.

11 David R. MacIver. Hypothesis. `https://hypothesis.readthedocs.io/`, 2018. [Online; accessed April 2020].

12 Rickard Nilsson. Scalacheck: Property-based testing for Scala. `https://www.scalacheck.org/`, 2018. [Online; accessed April 2020].

13 R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2019. URL: `https://www.R-project.org/`.

14 Frank Yates. Contingency tables involving small number and the χ^2 test. *Supplement to the Journal of the Royal Statistical Society*, 1(2):217–235, 1934.

[8] `http://www.pythontutor.com/c.html`