

Second International Computer Programming Education Conference

ICPEC 2021, May 27–28, 2021,
University of Minho, Braga, Portugal

Edited by

Pedro Rangel Henriques

Filipe Portela

Ricardo Queirós

Alberto Simões



Editors

Pedro Rangel Henriques 

Universidade do Minho, Portugal
prh@di.uminho.pt

Filipe Portela 

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

Ricardo Queirós 

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

Alberto Simões 

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

ACM Classification 2012

Applied computing → Education

ISBN 978-3-95977-194-8

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/978-3-95977-194-8>.

Publication date

July, 2021

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>.

License

This work is licensed under a Creative Commons Attribution 4.0 International license (CC-BY 4.0): <https://creativecommons.org/licenses/by/4.0/legalcode>.



In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:

- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/OASlcs.ICPEC.2021.0

ISBN 978-3-95977-194-8

ISSN 1868-8969

<https://www.dagstuhl.de/oasics>

OASlcs – OpenAccess Series in Informatics

OASlcs is a series of high-quality conference proceedings across all fields in informatics. OASlcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

Editorial Board

- Daniel Cremers (TU München, Germany)
- Barbara Hammer (Universität Bielefeld, Germany)
- Marc Langheinrich (Università della Svizzera Italiana – Lugano, Switzerland)
- Dorothea Wagner (*Editor-in-Chief*, Karlsruher Institut für Technologie, Germany)

ISSN 1868-8969

<https://www.dagstuhl.de/oasics>

Alfred Aho and Jeffrey Ullman
Turing Award Winners for 2021

for their work on the theory of compilers,
allowing us to use programming languages.

■ Contents

Preface	
<i>Pedro Rangel Henriques, Filipe Portela, Ricardo Queirós, and Alberto Simões</i>	0:ix–0:x
Committees	
.....	0:xi–0:xii
Authors	
.....	0:xiii–0:xv
Online-Teaching Environment with Gamification – A real Case Study	
<i>Filipe Portela</i>	1:1–1:13
Moving Classes in a Large Programming Course Online: An Experience Report	
<i>Hrafn Loftsson and Ásrún Matthíasdóttir</i>	2:1–2:13
Programmers’ Affinity to Languages	
<i>Alvaro Costa Neto, Cristiana Araújo, Maria João Varanda Pereira, and Pedro Rangel Henriques</i>	3:1–3:7
Melodic – Teaching Computational Thinking to Visually Impaired Kids	
<i>Rui Costa, Cristiana Araújo, and Pedro Rangel Henriques</i>	4:1–4:14
An Open-Source Gamified Programming Learning Environment	
<i>José Carlos Paiva, Ricardo Queirós, José Paulo Leal, Jakub Swacha, and Filip Miernik</i>	5:1–5:8
Integrating a Graph Builder into Python Tutor	
<i>Diogo Soares, Maria João Varanda Pereira, and Pedro Rangel Henriques</i>	6:1–6:15
Matching User Interfaces to Assess Simple Web Applications	
<i>Marco Primo and José Paulo Leal</i>	7:1–7:6
Active Methodologies in Incoming Programming Classes	
<i>João Paulo Aires, Simone Bello Kaminski Aires, Maria João Varanda Pereira, and Luís M. Alves</i>	8:1–8:9
Moopec: A Tool for Creating Programming Problems	
<i>Rui C. Mendes</i>	9:1–9:7
Automated Java Challenges’ Security Assessment for Training in Industry – Preliminary Results	
<i>Luís Afonso Casqueiro, Tiago Espinha Gasiba, Maria Pinto-Albuquerque, and Ulrike Lechner</i>	10:1–10:11
Exploring a Board Game to Improve Cloud Security Training in Industry	
<i>Tiange Zhao, Tiago Gasiba, Ulrike Lechner, and Maria Pinto-Albuquerque</i>	11:1–11:8
A System Architecture to Detect and Block Unwanted Wireless Signals in a Classroom	
<i>Daniel Barros, Paulo Barros, Emanuel Lomba, Vítor Ferreira, and Pedro Pinto</i> ..	12:1–12:7
A Teaching Assistant for the C Language	
<i>Rui C. Mendes and José João Almeida</i>	13:1–13:8



SHREWS: A Game with Augmented Reality for Training Computational Thinking <i>Francisco Saraiva, Lázaro V. O. Lima, Cristiana Araújo, Luís Gonzaga Magalhães, and Pedro Rangel Henriques</i>	14:1–14:10
Understanding Effects of the Algorithm Visualized with AR Techniques <i>Lázaro V. O. Lima, Manuel Sousa, Luis Gonzaga Magalhães, and Pedro Rangel Henriques</i>	15:1–15:10
Experiments on PR-Based Gamification <i>Alberto Simões and Ricardo Queirós</i>	16:1–16:10
User Experience Evaluation in a Code Playground <i>Ricardo Queirós, Mário Pinto, and Teresa Terroso</i>	17:1–17:9
Can I Code? User Experience of an Assessment Platform for Programming Assignments <i>Anne Münzner, Nadja Bruckmoser, and Alexander Meschtscherjakov</i>	18:1–18:12

■ Preface

The success of the 2020 edition of the 'International Computer Programming Education Conference (ICPEC)' impelled us to organize a new edition under the general topic of Computing at School. We actually believe that children shall be introduced to the computing world at primary school, and even younger people shall start earlier to train their skills to solve problems acquiring what is nowadays called *Computational Thinking*.

Four speakers from UK, Brasil, and Portugal were invited to talk about their motivations and experiences leading in their countries the referred international movement Computing at School.

Following the first edition of ICPEC, the second edition was once again online. We all know that a remote conference is always restrictive in terms of networking that is usually built at events of this kind. However, the typology of the event did not limit the participation among researchers and teachers on the main topic of the ICEPC, which is, the discussion on methodologies, trends and tools to improve the teaching-learning process of computer programming.

Inevitably, the papers in this second edition cover different approaches to overcome not only the complexity inherent to the field of computer programming, but also to mitigate the disadvantages of this new teaching paradigm caused by the pandemic. Many approaches are discussed in this conference, ranging from psychological studies to computer-mediated teaching tools including resources to aid children with special needs.

This book compiles 18 papers, accepted and revised for the 2.nd edition of ICPEC'2021, held virtually at Minho University, Braga, Portugal, from 27th to 28th of May.

The introduction of specialized services to automate tasks traditionally done manually by teachers or the inclusion of visualization mechanisms, playful design and gamification to involve students are the most discussed topics.

In the first case, the tendency is to integrate digital assistants or services in order to alleviate all (or part) of the manual phases of the teaching-learning process of computer programming. In this context, works related to tasks that typically are naturally time-consuming and error-prone are presented, such as the creation of programming exercises, program evaluation and feedback generation. It should be noted that in these topics there is a common point that concerns researchers and that relates to interoperability, not only in terms of data representation but also in the way the data is communicated between systems.

In the second case, several works related to game-based solutions are presented. To involve and motivate students in the computer programming domain, there are papers describing the use of visual feedback during the execution of programs, or the injection of gamification elements such as the use of leaderboards, achievements, badges and levels. To foster immersion, some authors propose the inclusion of virtual reality or the resort to serious games. Researchers consider that the use of approaches that inherit many concepts from games, should be applied sparingly so systems that use them do not transform into environments that are demotivating, unfair or that foster too many competitive facets which will hinder healthy and cooperative learning process.

Regardless of the approach proposed, the main objective of all these works is similar: *to motivate students to learn programming by promoting the practice supported by rich and immediate feedback.*

As ICPEC'2021 Chairs, we want to thank the many people without whom this event would never have been possible.



The invited Speakers (Simon Peyton-Jones, Sue Sentance, Christian Puhmann Brackmann, and Anabela Jesus Gomes) that let us learn with their researches and experiences; all the Members of the Scientific Committee for their valuable effort reviewing the submissions to support us in deciding the final list of accepted papers; all the Members of the Organizing Committee for looking carefully after all the details concerned with the logistics necessary to put up the event. Last but not the least, we express our acknowledgments to: the Authors that communicate their fully implemented ideas or projects, or their fresh proposals that are intended to be realized in the near future; and the Participants that actually made the conference happen and be a fruitful forum for the exchange of experiences and know-how.

Pedro Rangel Henriques

Filipe Portela

Ricardo Queirós

Alberto Simões

■ Committees

Conference Chairs

Pedro Henriques
Universidade do Minho, Portugal

Ricardo Queirós
Politécnico do Porto, Portugal

Alberto Simões
Politécnico do Cávado e Ave, Portugal

Filipe Portela
Universidade do Minho, Portugal

Alexandre Cardoso
Federal University of Uberlandia, Brazil

Ana Azevedo
Instituto Politécnico do Porto, Portugal

Anabela Gomes
Instituto Politécnico de Coimbra, Portugal

Antonio Manso
Instituto Politécnico de Tomar, Portugal

António Mendes
Universidade de Coimbra, Portugal

Steering Committee

Ricardo Queirós
Politécnico do Porto, Portugal

Alberto Simões
Politécnico do Cávado e Ave, Portugal

Mário Pinto
Politécnico do Porto, Portugal

Filipe Portela
Universidade do Minho, Portugal

Antonios Andreatos
Hellenic Air Force Academy, Greece

Bertil Marques
Instituto Politécnico do Porto, Portugal

Cristiana Araújo
Universidade do Minho, Portugal

Daniela Pedrosa
Universidade de Aveiro, Portugal

Dimitrios Koutsomitropoulos
University of Patras, Greece

Organizing Committee

Cristiana Araújo
Universidade do Minho, Portugal

Diana Barbosa
Universidade do Minho, Portugal

Goretti Pereira
Universidade do Minho, Portugal

Lázaro Lima
Universidade do Minho, Portugal

Paula Tavares
Instituto Politécnico do Porto, Portugal

Fernando Moreira
Universidade Portucalense, Portugal

Filipe Portela
Universidade do Minho, Portugal

Inna Skarga-Bandurova
Dahl East Ukrainian National, Ukraine

J. Ángel Velázquez-Iturbide
Universidad Rey Juan Carlos, Spain

Jakub Swacha
University of Szczecin, Poland

José Carlos Paiva
Universidade do Porto, Portugal

Program Committee

Alberto Simões
Instituto Politécnico do Cávado e Ave,
Portugal

Kostas Kolomvatsos
Universidade do Porto, Portugal

Leonel Morgado
Universidade Aberta, Portugal



Manuele Kirsch-Pinheiro
University of Paris 1, France

Marco Temperini
Sapienza Università di Roma, Italy

Maria João Varanda Pereira
Instituto Politécnico de Bragança, Portugal

Mário Pinto
Instituto Politécnico do Porto, Portugal

Miriam Antón-Rodríguez
Universidad Valladolid, Spain

Martinha Piteira
Instituto Politécnico de Setúbal, Portugal

Nikolaos Matsatsinis
Technical University of Crete, Greece

Paula Morais
Universidade Portucalense, Portugal

Paula Tavares
Instituto Politécnico do Porto, Portugal

Pedro Guerreiro
Universidade do Algarve, Portugal

Pedro Rangel Henriques
Universidade do Minho, Portugal

Pedro Ribeiro
Universidade do Porto, Portugal

Pedro Vasconcelos
Universidade do Porto, Portugal

Ricardo Queirós
Instituto Politécnico do Porto, Portugal

Rita P. Ribeiro
Universidade do Porto, Portugal

Roberto Hirata Jr.
University of São Paulo, Brazil

Rui Mendes
Universidade do Minho, Portugal

Sergio Ilarri
University of Zaragoza, Spain

Štefan Korečko
Technical University of Košice, Slovak
Republic

Teresa Terroso
Instituto Politécnico do Porto, Portugal

Vitor Sa
Universidade Católica Portuguesa, Portugal

Zuzana Kubincová
Comenius University of Bratislava, Slovakia

■ Authors

Alberto Simões

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

Alexander Meschtscherjakov

Center for Human-Computer Interaction,
University of Salzburg, Austria
alexander.meschtscherjakov@sbg.ac.at

Alvaro Costa Neto

Instituto Federal de Educação,
Ciência e Tecnologia de São Paulo
Barretos, Brazil
nepheus.br@gmail.com

Anne Münzner

Center for Human-Computer Interaction,
University of Salzburg, Austria
anne.muenzner@sbg.ac.at

Ásrún Matthíasdóttir

Department of Sport Science,
Reykjavik University, Iceland
asrun@ru.is

Cristiana Araújo

Centro Algoritmi
Departamento de Informática
Universidade do Minho, Portugal
decrisianaaraujo@hotmail.com

Daniel Barros

Instituto Politécnico de Viana do Castelo,
Portugal
danielbarros@ipvc.pt

Diogo Soares

University of Minho, Portugal
a74478@alunos.uminho.pt

Emanuel Lomba

Instituto Politécnico de Viana do Castelo,
Portugal
emanuellomba@estg.ipvc.pt

Filip Miernik

University of Szczecin, Szczecin, Poland
filip@flexile.io

Filipe Portela

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

Francisco Saraiva

Centro Algoritmi
Departamento de Informática
Universidade do Minho, Portugal
francisco_saraiva94@hotmail.com

Hrafn Loftsson

Department of Computer Science,
Reykjavik University, Iceland
hrafn@ru.is

Jakub Swacha

University of Szczecin, Szczecin, Poland
jakub.swacha@usz.edu.pl

João Paulo Aires

Departamento Acadêmico de Computação
Universidade Tecnológica Federal do Paraná
Brasil
joao@utfpr.edu.br

José Carlos Paiva

CRACS - INESC-Porto LA, Porto, Portugal
DCC - FCUP, Porto, Portugal
josepaiva94@gmail.com

José João Almeida

Centro Algoritmi
Departamento de Informática
Universidade do Minho, Portugal
jj@di.uminho.pt

José Paulo Leal

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

Lázaro Vinícius de Oliveira Lima

Centro Algoritmi
Departamento de Informática
Universidade do Minho, Portugal
lazarro.lima@ifb.edu.br



Luís Afonso Casqueiro

Instituto Universitário de Lisboa
(ISCTE-IUL), ISTAR, Lisboa, Portugal
luis_afonso_casqueiro@iscte-iul.pt

Luís Gonzaga Magalhães

Centro Algoritmi
Universidade do Minho, Portugal
lmagalhaes@dsi.uminho.pt

Luís M. Alves

Research Centre in Digitalization and
Intelligent Robotics (CeDRI)
Instituto Politécnico de Bragança, Portugal
lalves@ipb.pt

Manuel Sousa

Universidade do Minho
Braga, Portugal
manuelgcsousa@gmail.com

Marco Primo

Faculty of Sciences,
University of Porto, Portugal
up201800388@edu.fc.up.pt

Maria João Varanda Pereira

Research Centre in Digitalization and
Intelligent Robotics (CeDRI)
Instituto Politécnico de Bragança, Portugal
mjoao@ipb.pt

Maria Pinto-Albuquerque

Instituto Universitário de Lisboa
(ISCTE-IUL), ISTAR, Portugal
maria.albuquerque@iscte-iul.pt

Mário Pinto

uniMAD - ESMAD, Polytechnic of Porto,
Porto, Portugal
mariopinto@esmad.ipp.pt

Nadja Bruckmoser

University of Salzburg, Austria
nadja.bruckmoser@stud.sbg.ac.at

Paulo Barros

Instituto Politécnico de Viana do Castelo,
Portugal
paulobs@ipvc.pt

Pedro Pinto

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

Pedro Rangel Henriques

Centro Algoritmi
Departamento de Informática
Universidade do Minho, Portugal
prh@di.uminho.pt

Ricardo Queirós

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

Rui Costa

Centro Algoritmi
Departamento de Informática
Universidade do Minho, Portugal
rui.diogo.costa@hotmail.com

Rui Mendes

Centro Algoritmi,
Departamento de Informática,
Universidade do Minho, Portugal
azuki@di.uminho.pt

Simone Bello Kaminski Aires

Departamento Acadêmico de Computação
Universidade Tecnológica Federal do Paraná
Brasil
sbkaminski@utfpr.edu.br

Teresa Terroso

uniMAD - ESMAD, Polytechnic of Porto,
Porto, Portugal
teresaterroso@esmad.ipp.pt

Tiange Zhao

Siemens AG, Munich, Germany
Universität der Bundeswehr München,
Germany
tiange.zhao@siemens.com

Tiago Gasiba

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

Ulrike Lechner

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

Vítor Ferreira

Instituto Politécnico de Viana do Castelo,
Portugal
ferreira@estg.ipv.c.pt

Online-Teaching Environment with Gamification – A real Case Study

Filipe Portela  

Algoritmi Research Centre, University of Minho, Braga, Portugal

IOTech – Innovation on Technology, Trofa, Portugal

Abstract

Teaching processes are changing, and Higher Education is not an exception. Professors are adapting their teaching methods to b-learning classes. They are looking for innovative approaches and tools that allow engaging students in the classrooms. This paper presents the results of a teaching and challenging experience. The TeachTeach paradigm was used in a fully online environment. Professors explored several methods/approaches during the semester, and the students were faced-off with a new reality of learning. They attested students skills like programming, resilience, innovation and entrepreneur capabilities during the development of a project with an actual web application. The results show that the efforts were tremendous but beneficial for all the stakeholders. At the end of the case study, a few numbers should be highlighted: 11,173 Downloads, 15,224 Messages, 200,000 Sessions and 208 online hour classes. Comparing this approach with other curricular units (CUnit) online strategies, 96,53% of the students considered it equal or better.

2012 ACM Subject Classification Applied computing → Interactive learning environments

Keywords and phrases TechTeach, Case Study, Online-Learning, ioEduc, Computer Programming, Gamification, Innovation

Digital Object Identifier 10.4230/OASlcs.ICPEC.2021.1

Acknowledgements I want to thank IOTECH for supporting the project.

1 Introduction

Nowadays, Education is facing a significant digital transformation process. Universities are experiencing a set of essential changes induced by technological and social trends towards digitisation. These changes were aggravated by Coronavirus disease (COVID-19) [1]. COVID-19 brought new challenges to Education where professors and schools had to take the lead of this unexpected digital transformation without being prepared for it [12]. Higher Education is not an exception and, unfortunately, most professors migrated their presential classes to an online schema without them being transformed/adapted. Several techniques and approaches are being explored, but the teaching and assessment methods are not accompanying the changes, and professors limited their action to transpose their teaching mechanisms to distance classes.

Conscientious of this fact, a new and innovative teaching approach was proposed. TechTeach is a disruptive approach in which the focus is the students and their motivation. This approach was designed for several years and showed good results in a presential contact environment; however, two critical questions arise now: “Is TeachTeach ready or adapted to classes 100% online?”. “What is the effort needed? Is it worth it?”

This paper will answer both questions after dissecting the experience performed during a semester. All the classes and assessment methods were conducted online, using sync and async activities. This experience was a challenge for everyone, and the results were promising. This article also shows some ideas about converting presential classes to B-Learning approaches with sync and async tasks. In fact, this paper presents a proof of concept of a case study on a Programming Curricular Unit (CUnit) with ten (10) European



© Filipe Portela;
licensed under Creative Commons License CC-BY 4.0

Second International Computer Programming Education Conference (ICPEC 2021).

Editors: Pedro Rangel Henriques, Filipe Portela, Ricardo Queirós, and Alberto Simões; Article No. 1; pp. 1:1–1:13



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

1:2 Online-Teaching Environment with Gamification

Credit Transfer and Accumulation System (ECTS) with almost one hundred and seventy (170) students. Innovation and entrepreneur skills are applied through the development of a real-life case project and exploration of soft-skills.

The paper is divided into six sections: after a brief introduction, the paper background is explained. Then, section three presents all the methods and tools used in this case study. The case study is described in section four, and the achieved results are analysed in the discussion section. Finally, a conclusion of the work is provided.

2 Background

This section helps to understand the case study theoretical context better by reviewing some concepts.

2.1 Online Classes

Online learning has several characteristics that impact faculty implementation and CUnit progress [9]. One of the most significant changes in Education is the classroom converting from presential to online. According to a study [2], online and remote learning in higher education institutes are a necessity in times of COVID. Wahab Ali [2] also mentioned that the massive technology advance represents a shift in educational goals and aspirations. Online classes can have better results with bigger classes [16]; however, the typical conversion of teaching environments is not the best solution.

2.2 Digital Transformation at Classrooms

Digital transformation can bring new knowledge and practices of teaching, learning, communicating, and organising schoolwork [18]. A study [14] showed that the ability to have a digital competence is identified by someone who can use laptops and various digital learning resources in a positive way. In this aspect, the same study highlighted that the context/environment determine professors' digital competences. Their decisions are based on their own value frameworks and approaches.

2.3 Related Works

This work's main goal is to do a proof concept of TeachTeach and not address or compare it to other teaching methods. Even though there are other similar works in this area, none was considered.

3 Material and Methods

This case study explored TechTeach and all the related approaches.

3.1 TechTeach

TechTeach uses Blended-Learning, Gamification, Soft-skills, Quiz and Surveys, Flipped classrooms and Project Base Learning to proportionate the best learning environment for the students [20]. According to the author [21], TechTeach can be adapted to different lessons and environments. The case study explores this method in a non presential space and proves its adequacy for online teaching environments.

3.1.1 Project Based Learning

Project-based learning (PBL) is an approach to learning [4] and explores the use of exercises to proportionate practical learning. PBL is a student-centred pedagogy [10] and involves a dynamic classroom approach. Professors use the active exploration of real-world challenges and problems to stimulate students to achieve new knowledge [10].

3.1.2 Gamification

According to Kapp et al. [13], Gamification consists of “using game-based mechanics, aesthetics and game thinking to engage people, motivate action, promote learning, and solve problems”. Learning environments with gamification activities can have a positive impact on the learning outcome [7]. In Education, Gamification is being used to encourage students to perform specific tasks; however, the lack of a narrative makes it difficult to measure the engagement influence [15].

3.1.3 Flipped Classes

Flipped learning is an instructional approach used to support teaching [7]. This type of learning allows students to have their own learning environments outside school [22]. The classes using this method are different and more practical because professors can use the contact time to discuss the most relevant topics and clarify students doubts instead of having monotonous exposing classes.

3.1.4 Skills

Skills achievement is a natural process of human being development [3]. The constant changes in the modern world require more flexibility and more vital adapting skills [17]. According to Araujo [3], this concept is also developed in the educational scope and elevates the cognitive and constructionist pedagogical perspectives to the detriment of behaviourism. In the case study, the two types of skills were trained: technical (web programming and derivatives) and soft (resilience, innovation, argumentation, communication, leadership, entrepreneurship, among others).

3.2 Tools

This subsection does a brief overview of the tools used in this case study used to apply TechTeach.

3.2.1 Teach Supporting

ioEduc is an owner tool of IOTech and arises due to the increasing use of mobile devices at the classroom [23]. This platform was designed to support teachers and students in their tasks during classes [23]. Its web-based features (e.g. attendances, chat, slides, quizzes, surveys, drive, FAQs, peer-assessment) make it an excellent tool to support online classes.

ioChat arised as an internal communication platform of IOTech. Its success near the costumers motivated the Team to explore it in a different context such as Education. ioChat is based on an open-source software – RocketChat. It is fully customised and has many features like messages/conversations (text, audio and video), rooms (grouped, private, public or discussion), notifications (online, email, push), plugins (drive, calendar, polls), among others.

1:4 Online-Teaching Environment with Gamification

Zoom is currently one of the most used video-conferencing platforms. Their robustness allows people to maintain high-quality meetings with a vast number of participants. Zoom allows creating an online teaching environment due to its functionalities. It can connect long-distance participants across rooms systems, desktops and mobile devices to seamlessly bring together from various places [25].

3.2.2 Collaborative and Competitive

Kahoot! is characterised as a response system that engages participants through game-like pre-made or impromptu quizzes, discussions, and surveys [8]. Its game-based learning allows professors to support their education activities [5]. According to Carolina et al. [19], “Instructional games are gaining acceptance in the classroom as the eLearning merits of student engagement, and immediate feedback is recognised.”

HackerRank [11] can be classified as a classic competitive programming platform [6] with several different types of exercises (e.g. problem-solving or challenges).

3.2.3 Data Analytics

Google Analytics is a web analytical tool that helps to analyse websites’ traffic [24]. This tool is essential to understand the tools/websites’ impact/usage.

4 Case Study

The following subsections characterise the case study – Application of TeachTeach on a computer programming unit at the University of Minho to create an entire online learning environment.

4.1 Context

To better understand the case study, it is essential to analyse the context/environment of this Curricular Unit (CUnit):

- Integrated Master degree;
- 2nd year of study plan;
- Ten European Credit Transfer and Accumulation System (ECTS);
- Online from 5 of October 2020 and 29 of January 2021;
- One hundred and sixty-eight students registered (90% participants);
- Five professors;
- Three types of classes: Theoretical (T), Theoretical-practice (TP), laboratory practices (LP);
- Fifteen weeks of work;
- Thirteen weeks of contact classes;
- Fourteen hours of work by week/student;
- Six hours of contact by week;
- Two Hours of Tutorial Working Time (OT).

Before the CUnit starts, the coordinator professor defined two main goals according to the TechTeach paradigm:

1. To Innovate and guarantee the success of a 100% Online CUnit;
2. To use a set of concepts/trends: Gamification, PBL, Quizzes, and Flipped Classes and emerging tools to provide students’ best online learning environment.

The working/teaching plan of Web Programming CUnit was based on the following assumptions:

- Full online classes (100%);
- Gamification mechanisms to assess students' performance;
- Online communication platform that allows simulating the classroom environment;
- Tests and assessments – dynamic, inclusive and online;
- Preparation of learning content/tutorials to be carried out outside of classes (OT);
- Online repository (drive) with complementary information (books, articles, videos or tutorials);
- FAQs with the most common issues;
- Recording of theoretical classes;
- Project-based learning with the application of a real case;
- Room for doubts and online chat;
- Continuous assessment of the CUnit – direct contact with the class delegates;
- Exploration of soft-skills (e.g. entrepreneurship, cross-learning, communication, adaptability, resilience and leadership).

4.2 Approach

TechTeach was designed to be used by any teacher who wants to innovate and have a different teaching view. This CUnit has a strong focus on the project component. So, to ensure the correct implementation of this mechanism, professors used several approaches.

4.2.1 Classes

All the classes ran online, and three tools were used:

Zoom – At the beginning of each theoretical class, the professor started the recording and shared the ioEduc live system (containing the slides and other features) in Zoom. After a brief explanation of the week's main topics, the professor created several breakout rooms, and the students were randomly distributed. Students were faced with solving some challenges in a group according to the topics addressed in this phase. They can ask for support here or in ioChat, and professors were jumping between each room. Each room was an average of 5-6 students, i.e., each class had 20 rooms on average. Before classes ending, all the students returned to the main room, and the professor performed a raffle among the attendees to determine which students have a bonus in the weekly quiz.

ioChat – ioChat was the main communication tool. The professor created several conversation rooms according to classes, teams and groups. All the students and professors were registered in ioEduc and had access to this communication tool. At the beginning of each class (TP and LP), the professor started a video-call with all of the students. The Professor used this first video call to talk a little about the class goals. Then, all the students went to their particular room and initiated another video-call. Professor visited each room to explain possible doubts and to give support in the project development. This tool allowed professors to have parallel classes on the same platform. In the TP classes, students were grouped by Team according to their role (front-end, back-end and full-stack). In the LP classes, the students were organised by group, following the allocation of figure 2 1. ioChat also had a set of creation rules (rooms name, username, photo, among other details) to quickly identify students, groups, projects, and professors. This tool was also used to promote and facilitate the communication inside and outside of the classes between all the stakeholders.

1:6 Online-Teaching Environment with Gamification

ioEduc – ioEduc was the centre of everything. All the slides were available in this platform; students can, for example, ask questions, answer the quizzes, consult FAQs or perform a peer-assessment. In all classes the attendances are registered at ioEduc through the attendances system. ioEduc' drive had T recorded classes and many books, tutorials, and videos to support the OT work. Regarding the professors, the attribution of cards was done here as well. ioEduc also has a connection to many complementary tools useful to help the classes (e.g. Cloud 9 AWS, GitHub or Heroku).

4.2.2 Project

The project developed in this CUnit tried to bring some innovation and entrepreneurship to the entire process and asked students to propose a solution to a real case. In a brief overview of the project statement, it is important to mention the following points:

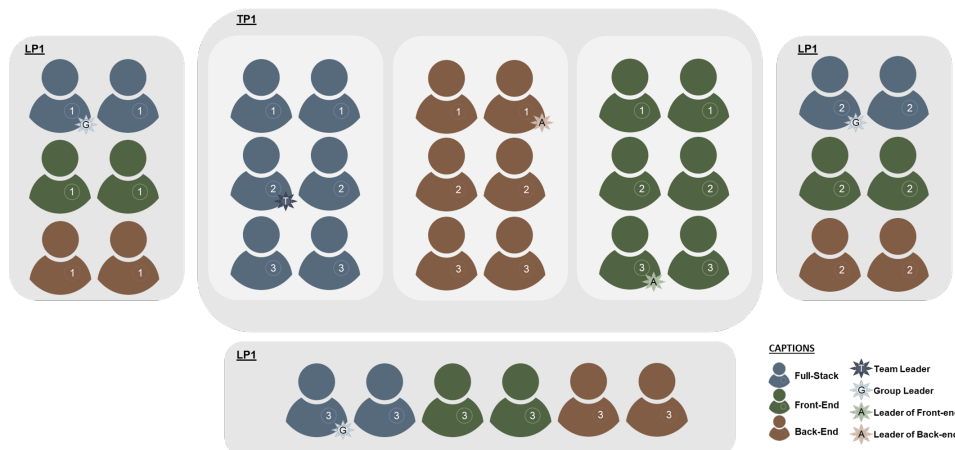
1. The company “Secure4All” is responsible for the coordination of operations. It needs a web application that helps managing each of the occurrences received, and a web / mobile application that allows the interaction of operations on the field with their occurrences.
2. The intention was to develop a web/mobile application/system (PWA) that supports a public intervention unit's management.
3. In an initial phase, Secure4All defined a strategy to digitise processes for some of its intervention units, namely: Firefighters (Fire), PJ (Cyber-Attack), GNR (Events), Municipal Police (Assaults), ASAE (Food Inspection), INEM (Accident), Maritime Police (Drowning), and PSP (Disturbances).
4. The teams selected the project themes, and each Team was composed by three groups. Each group was responsible for developing a set of features/components.
5. The Team gathered and defined the components for each group. Then, each group was responsible for developing the entire Front-end (visual) and Back-end (Server) layer of the assigned component solution.
6. ioChat was the communication platform between all the stakeholders.

Regarding the team, the distribution was the following:

- Group I** – Operations Center;
- Group II** – Operations manager / Administrators;
- Group III** – Field Operations Officer;
- Group IV** – Audit (external entity).

To better comprehend students distributions by each class type, figure 1 presents how the teams and groups were organised.

The teams were constituted according to the TPs shifts. Each TP shift had two teams with a specific theme, and each Team had three or four groups. The groups were composed by 6 six elements (+/- 1 depending on the number of students in the shift). Each group was composed by at least 2 Front-End students, 2 Back-End students, and 1 Full-Stack student. Each element was responsible for the development of a set of functionalities/interactions. Each group' functionalities was divided between front-office (without login) and back-office (with login). Each Team created the terms of reference plan at the begin of the project. This document included project requirements, distribution of roles and tasks by the group and members, and a projected project cost (i.e., the grade that they want to achieve if they meet all the requirements). The terms of reference represents their commitment to the client (teaching team) and must show their innovative and entrepreneurial vision to the project.



■ **Figure 1** Teams and Groups distribution retrieved from [21].

4.2.3 Assessment

This CUnit has a transversal and continuous assessment method that evaluates students, professors, and the respective unit. The students' assessment was divided into three types:

- A weekly quiz with a bonus to award motivation, work, and participation;
- Project with the cards system to value skills;
- Mini-tests with three synchronised methods to assess different parts of knowledge (Front-end, Back-end and Full-stack).

The CUnit and professors were also evaluated in two moments in the middle and the end of the semester. This assessment was used to assess several factors like professors knowledge, teaching skills, CUnit plan, and classes.

4.2.4 Gamification

Gamification was applied in several moments of the CUnit, and professors attributed positive and negative points during the classes. A narrative was designed for each method, and students knew from the beginning what they had to do in order to achieve specific goals. Gamification was applied in the following exercises/contexts:

- MT1 and MT3 – Grades were calculated using a metric based on the student knowledge, questions weight, and answering time according to the average time of respondents (3rd quartile).
- MT2 – Exercise 1 had code gaps, and students must complete it with their personal data. Then students had to use their skills/knowledge to unlock exercise 2.2. In the end, students achieved one of three possible grades:
 - -1 Student does not know the basics and should reprove to this component – 0%;
 - 0 Student knows the basic and can advance – 100%;
 - 1 Student knows the basics and overcomes all the challenges (e.g. unlock the 2nd part of the exercise) and must advance – 110%.
- The final grade of MT was achieved by the formula $\text{avg}(\text{MT1}, \text{MT3}) * \text{MT2}$.
- Quiz with bonus – The students had the opportunity to duplicate their weekly quizzes results. The Professor performed every T classes a selection of 10 to 15 students. The rules were simple: only the attendants of that T class were eligible, and no one could be selected more than two times without all the students being selected at least one time.

1:8 Online-Teaching Environment with Gamification

In the end, the student with the most points without a bonus (K) had a grade of 20. All the students with points with a bonus higher than the K had also 20; then all the other students had a relative grade regarding K. The goal of this exercise was to motivate and award students who could hit more questions.

- Cards System – The card system was applied during the project. Team leaders and professors attributed yellow (negative) or white cards (positive) to the students.
- Rescue System – This system was only available in MT2 to the students who had -1 grade and considered it unfair. In this situation, professors analysed the student case and, in case of acceptance, they allowed him to continue with a penalty of fifteen per cent (15%) in the final MT grade (i.e., MT2 equal to 85%).

4.3 Numbers

The following numbers help to understand the impact and the effort needed to have a CUnit with a complete online-learning environment:

- 13 shifts (1T, 4TP, 8LP);
- 1 Real project with three evaluation phases;
- 8 teams (+/- 20 students) and 27 groups (+/- 6 students);
- 3 synchronous Mini-Tests (2 Kahoot, 1 HackerRank);
- 2 CUnit and professors evaluations (middle and end);
- 115 was the average number of responses in CUnit final evaluation;
- 85% of students approved the CUnit gamification system;
- 80% was the minimum attendance of T Classes;
- 144 was the number of bonus attributed;
- 21 cards (Blue, White, Yellow and Red) were attributed;
- 3 students activated the rescue system.

In this analysis, it is essential to highlight some of the achieved online results:

- 208 (two hundred and eight) hours online classes with parallel sessions, in a total of 338 (three hundred and thirty eight) hours;
- 80 (eighty) chat rooms;
- ~ 220k page views;
- ~ 25k access to the CUnit support platform (sessions);
- ~ 16k access to the chat (sessions);
- ~ 15k messages exchanged in the chat;
- ~ 11k file downloads;
- ~ 106 uninterrupted days online (sum of all active sessions time).

A visual analysis of the CUnit impact on the students' online usage and assessments is presented in figures 2 to 8. Figures 2 to 5 resume some of the numbers that are possible to achieve from ioEduc and ioChat (such as page views, events, sessions, among others). Regarding Figures 6 to 8, students evaluated TechTeach approach and their Gamification and online component. These figures prove the results mentioned in the before list and sustain the assumptions taken with this study.

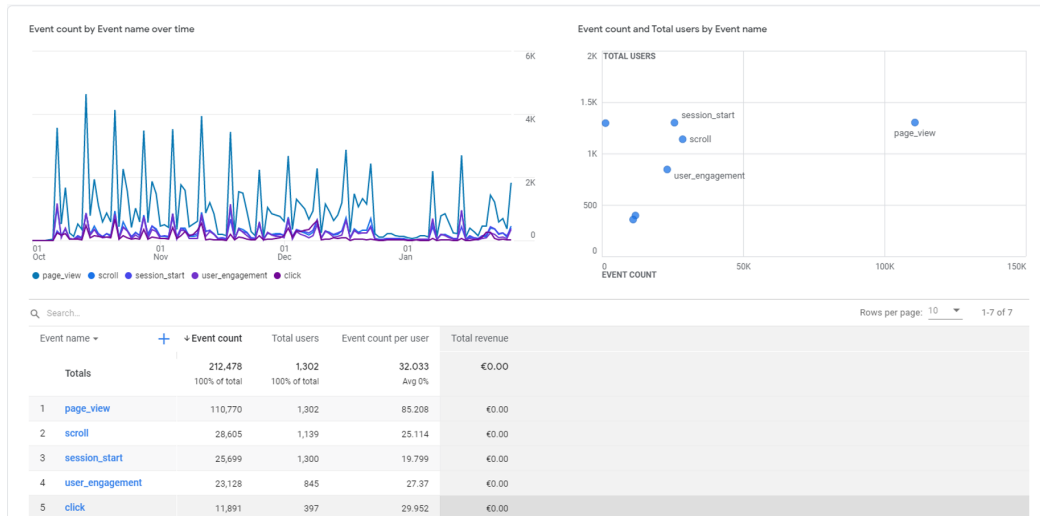


Figure 2 ioEduc – Session and Events.

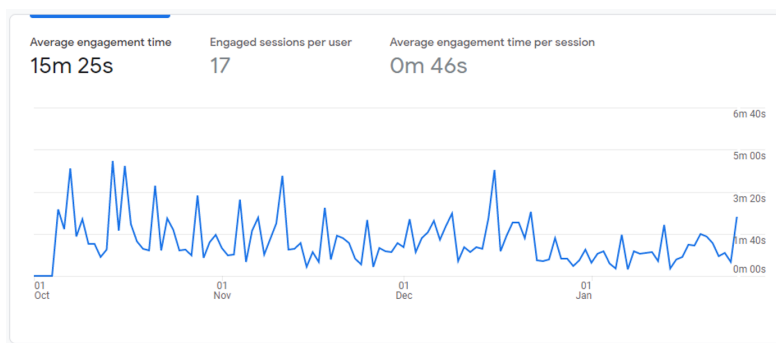


Figure 3 ioEduc – Session Average Time.

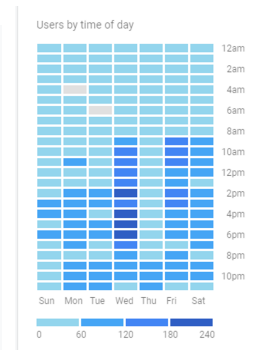


Figure 4 ioEduc – Users work/time.

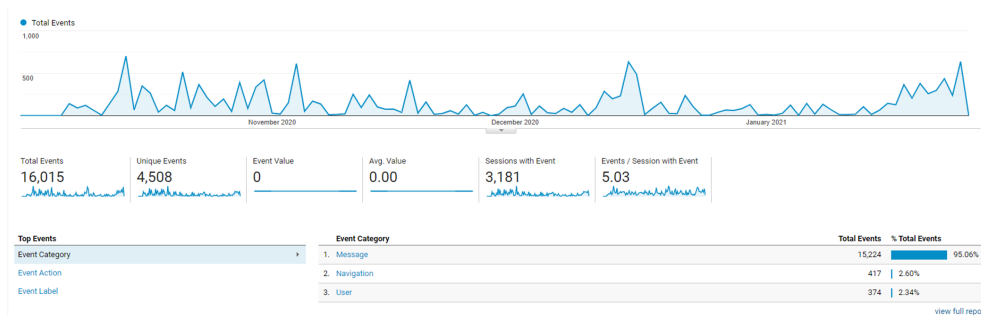
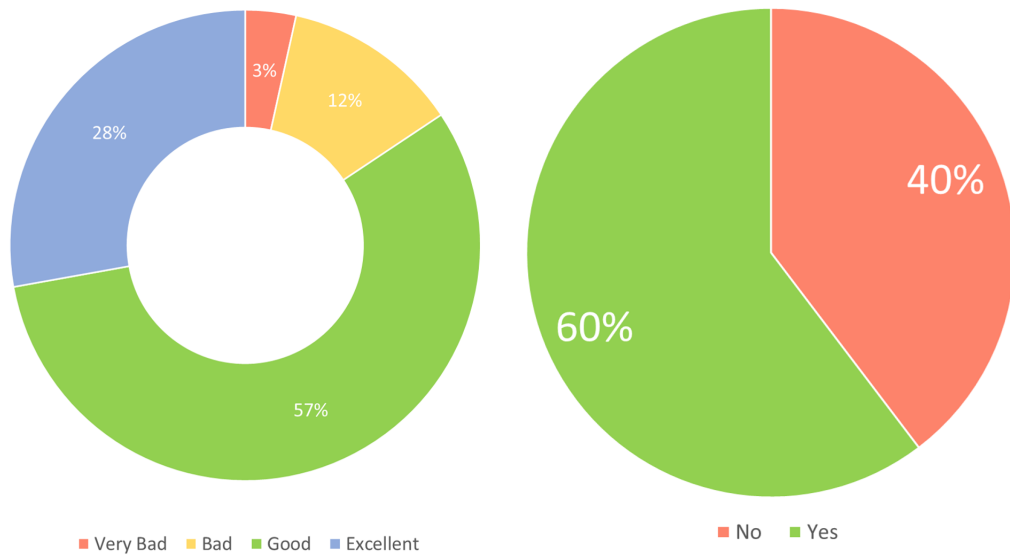
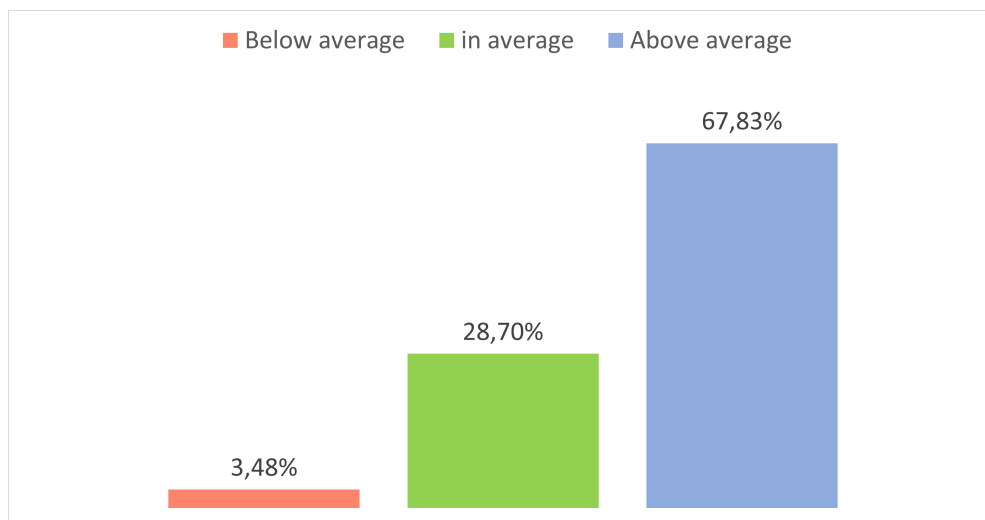


Figure 5 ioChat – Events.

1:10 Online-Teaching Environment with Gamification



■ **Figure 6** How was the gamification approach? ■ **Figure 7** Did you like having online classes?



■ **Figure 8** How do you classify this approach (TechTeach) when comparing it with other CUnit?

4.4 Word Cloud

The bag of words is an excellent technique to present students' opinions. After receiving the students' comments, the most positive words figure (9) – dynamic classes and professors – and negative aspects figure (10) – much mater and materials – were identified, as can be observed in the figures.



■ **Figure 9** Positive aspects of CUnit.



■ **Figure 10** Negative aspects of CUnit.

5 Discussion

After concluding this case study, the achieved numbers can give a detailed overview of the students and professors interactions and engagement. There were more than two hundred thousand (232,433) page views, and the average session time was approximately fifteen minutes (15m25s). Users exchanged more than fifteen thousand messages (15,224), and eleven thousand (11,173) downloads were made. A particular fact to mention is the working time: only two periods of time did not have anyone online – Monday from 4 am to 5 pm, and Tuesday from 5 am to 6 am. In all the other hours/day, at least one user was online. Both platforms have at least fifteen thousand accesses during the semester.

In terms of the mini-tests, the strategy used was to have simple exercises under pressure (without much time to think), and questions containing personal attributes that contributed to making each exercise unique. Generally, students liked the MT1 & MT3 (62.39 % approved the model) but disliked MT2. Both results showed the success of the strategy. MT1 and MT3 were easy and allowed to understand the basics, and MT2 was challenging to do copy or thinking; because of that, the students did not liked it.

Finally, and regarding the CUnit, students' opinion was very positive and motivating. More than ninety-five per cent of students (96,53%) considered this CUnit on the average or higher than average compared with other CUnits in this time of COVID-19. In terms of the online class question, it is essential to mention that, although most of the students (60%) liked it, 40% answered “no” The negative answer was essentially due to working conditions (e.g. one computer at home to many users/students or lousy internet).

On the positive side is the Teaching Team and CUnit and TechTeach as one whole, focusing on the professors' availability and dynamism. On the negative side, many answers were “nothing” to mention; however, students indicate some negative aspects related to the quantity of matter and materials taught in the Theoretical classes. Unfortunately, this issue has years and happens because this CUnit is the only one that addresses this thematic in the entire course. So the time is not enough to address all topics, but it is not possible to reduce more because it is crucial to give the students a global overview of Web Programming.

6 Conclusion

This work was used as proof of concept and the case study explored allowed professors to obtain an answer to the research question: “Is TeachTeach ready or adapted to classes 100% online?”. The answer is Yes. Both stakeholders (students and professors) considered this approach a success (>80% positive answers). Although good results achieved – more than 220k page views, 40k sessions, 11k downloads of files and 15k messages – it is vital to highlight the effort need to put into practice, for example, the coordinator’s effort was ~ 250% multiplied by their teaching hours (8) in an average of 20h a week. With this study, it was also possible to conclude that it is essential to have some “free” time to apply this methodology and keep students engaged. It is necessary to make an extra effort, a good teaching team and, as the word cloud shows, many availabilities to help the students. TechTeach is adequate to the constantly changing teaching system; it brings fresh air to this type of classes and has a favourable acceptance. Unfortunately, this study also showed a considerable gap. In some cases, and although the results are motivating, students fear and distrust online classes. On the other side, the limitations are notorious and, from the 40% of students who did not liked to have online classes, 80% considered that they do not have the correct conditions to do so.

Making a balance and after analysing students opinion (in a time of COVID-19, >95% of students considered this CUnit equal or better than the others online CUnit) and their grades (95% of the active students were approved), the results are much rewarding and offset the effort. As the main contributions of this work are:

- A guide of strategies/tools needed to have an entire online class;
- An idea of the effort needed to have this type of classes;
- Case study results and the receptivity of students to the TechTeach approach.

With this case study, it was possible to prove that the TechTeach approach can motivate students skills and promote innovation and entrepreneurship of them at a distance (online).




As future work, all the data will be dissected, and a Web Mining process will be applied to try to correlate the students’ performance with their activity.

References




- 1 Emilio Abad-Segura, Mariana-Daniela González-Zamar, Juan C. Infante-Moro, and Germán Ruipérez García. Sustainable management of digital transformation in higher education: Global research trends. *Sustainability*, 12(5), 2020. doi:10.3390/su12052107.
- 2 Wahab Ali. Online and remote learning in higher education institutes: A necessity in light of covid-19 pandemic. *Higher Education Studies*, 10(3):16–25, 2020.
- 3 C.V. Araújo. The bologna process and curricular changes at higher education: what are skills for? *Educação e Pesquisa*, 44, 2018.
- 4 Stephanie Bell. Project-based learning for the 21st century: Skills for the future. *The clearing house*, 83(2):39–43, 2010.
- 5 Marisa Correia and Raquel Santos. Game-based learning: The use of kahoot in teacher education. In *2017 International Symposium on Computers in Education (SIIE)*, pages 1–4. IEEE, 2017.
- 6 Alexandru Dan and Ass Pr PhD Iftene Adrian. Computer science platform, 2018. BACHELOR THESIS.
- 7 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.
- 8 Ryan Dellos. Kahoot! a digital game resource for learning. *International Journal of Instructional technology and distance learning*, 12(4):49–52, 2015.

- 9 Amber D Dumford and Angie L Miller. Online learning in higher education: exploring advantages and disadvantages for engagement. *Journal of Computing in Higher Education*, 30(3):452–465, 2018.
- 10 Glavin. Examples of project-based learning | k12 academics. <https://www.k12academics.com/Educational%20Practices/Project-Based%20Learning/examples-project-based-learning>, 2019. (Accessed on 03/09/2021).
- 11 HackerRank. Hackerrank. <https://www.hackerrank.com/>. (Accessed on 03/15/2021).
- 12 Netta Iivari, Sumita Sharma, and Leena Ventä-Olkkonen. Digital transformation of everyday life – how covid-19 pandemic transformed the basic education of the young generation and why information management research should care? *International Journal of Information Management*, 55:102183, 2020. Impact of COVID-19 Pandemic on Information Management Research and Practice: Editorial Perspectives. doi:10.1016/j.ijinfomgt.2020.102183.
- 13 K.M. Kapp. *The gamification of learning and instruction: Game-based methods and strategies for training and education*. John Wiley & Sons, 2012.
- 14 Anders D Olofsson, Göran Fransson, and J Ola Lindberg. A study of the use of digital technology and its conditions with a view to understanding what ‘adequate digital competence’ may mean in a national policy initiative. *Educational studies*, 46(6):727–743, 2020.
- 15 Paula Toledo Palomino, Armando M Toda, Wilk Oliveira, Alexandra I Cristea, and Seiji Isotani. Narrative for gamification in education: why should you care? In *2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT)*, volume 2161, pages 97–99. IEEE, 2019.
- 16 Marc Parry. Online, bigger classes may be better classes. *Chronicle of Higher Education*, 57(2):A1–A22, 2010.
- 17 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.
- 18 Fanny Pettersson. Understanding digitalization and educational change in school by means of activity theory and the levels of learning concept. *Education and Information Technologies*, 26(1):187–204, 2021.
- 19 Carolyn M Plump and Julia LaRosa. Using kahoot! in the classroom to create engagement and active learning: A game-based technology solution for elearning novices. *Management Teaching Review*, 2(2):151–158, 2017.
- 20 Filipe Portela. A new and interactive teaching approach with gamification for motivating students in computer science classrooms. In *First International Computer Programming Education Conference (ICPEC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
- 21 Filipe Portela. Techteach—an innovative method to increase the students engagement at classrooms. *Information*, 11(10):483, 2020.
- 22 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.
- 23 Miguel Silva, Diogo Ferreira, and Filipe Portela. Ioeduc - bring your own device to the classroom. In Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões, editors, *First International Computer Programming Education Conference, ICPEC 2020, June 25-26, 2020, ESMAD, Vila do Conde, Portugal (Virtual Conference)*, volume 81 of *OASICs*, pages 23:1–23:9. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/OASICs.ICPEC.2020.23.
- 24 Bill Su. What is google analytics, and why is it important to my business? | by bill su | analytics for humans | medium. <https://medium.com/analytics-for-humans/what-is-google-analytics-and-why-is-it-important-to-my-business-8c083a9f81be#>, 2017. (Accessed on 03/16/2021).
- 25 Jane Sutterlin. Learning is social with zoom video conferencing in your classroom. *ELearn*, 2018(12), 2018.

Moving Classes in a Large Programming Course Online: An Experience Report

Hrafn Loftsson   

Department of Computer Science, Reykjavik University, Iceland

Ásrún Matthíasdóttir   

Department of Sport Science, Reykjavik University, Iceland

Abstract

We present an experience report on moving face-to-face classes in a large CS1 course to an online format, during the COVID-19 pandemic. The course is based on the flipped classroom approach and team-based learning. Students prepare for classes by reading specific chapters of the textbook and/or by watching pre-recorded videos. The classes are synchronous, in which students take quizzes and work on programming assignments in teams, with the guidance of tutors. To evaluate the implementation, we compared the results from surveys and exams between 2019 and 2020. The results show that students were at least as satisfied with the online classes in 2020 in comparison with face-to-face classes from the previous year, and generally satisfied with the organization of the course and the learning experience. Moreover, we found no discernible change in the grades on the midterm exams and the final exam between the two years. In the future, we might allow the students to choose the class format that best fits their individual needs.

2012 ACM Subject Classification Applied computing → Education

Keywords and phrases online learning, flipped classroom, team-based learning, introductory programming, CS1, COVID-19

Digital Object Identifier 10.4230/OASICS.ICPEEC.2021.2

1 Introduction

During the last decade, teaching, learning, and collaborative approaches have become important topics in introductory programming education research [2]. The flipped classroom (FC) is a teaching approach in which the emphasis is on active learning. In the FC, students are expected to study specific course material outside the class, and then engage in learner-centered activities in the classroom with the help of (a) tutor(s) [1, 4, 12, 17, 18, 19, 22]. Active learning often involves Team-Based Learning (TBL) (or Project-Based Learning (PBL)) where students work on projects in a collaborative manner in groups [8, 9, 20].

Overall, the last couple of years have been very stressful, with extra workload for university faculty and staff, as well as for the students. The COVID-19 pandemic has affected the entire educational system world-wide, and most face-to-face courses at the university level were at one point moved to an online format, often within a short notice.

In this paper, we present a detailed experience report about moving face-to-face classes in a large CS1 course, which was already based on FC and TBL, to an online format. Our motivation for moving the classes online was purely due to necessity, i.e. due to the COVID-19 pandemic. However, moving classes to an online format can be beneficial, in general, as pointed out by Irani and Denaro [7]: “The flexibility of an online class makes it easier for students to schedule the coursework around other commitments, and commuting students can save time in not traveling to campus for every class”.

Our online classes are *synchronous* in the sense that students and teachers meet in real-time, with the help of a video-conferencing platform, at specific slots in the time table. This mode of classes, which has also been called “online face-to-face” [6], is in contrast to *asynchronous* online learning which does not require students and teachers to be online at the same time [19].



© Hrafn Loftsson and Ásrún Matthíasdóttir;
licensed under Creative Commons License CC-BY 4.0

Second International Computer Programming Education Conference (ICPEEC 2021).

Editors: Pedro Rangel Henriques, Filipe Portela, Ricardo Queirós, and Alberto Simões; Article No. 2; pp. 2:1–2:13



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

Our surveys show that students were at least as satisfied with the online classes in comparison with face-to-face classes from the previous year. Moreover, we found no discernible change in the grades on the midterms and the final exam between the two years.

We believe that the work described in this paper can be beneficial to teachers of CS1 courses who want to apply FC and TBL in their online courses.

This paper is organized as follows. We discuss related work in Section 2, present background for our work in Section 3, and the moving of our classes online in Section 4. The results and discussion regarding student surveys and exams are presented in Section 5 and 6, respectively. Finally, a conclusion is presented in Section 7.

2 Related work

2.1 The Flipped Classroom

Learning programming can be difficult for many students, which often leads to high failure and drop-out rates from programming courses [15]. Consequently, several CS departments have experimented with using the FC teaching method in introductory programming courses. In the FC, the “traditional” lecture is replaced with in-class activities and students are supposed to come prepared for class by watching pre-recorded videos and/or reading given text material. In what follows, we briefly review a few of the recent papers that have shown benefits of using an active learning approach like FC in introductory programming courses.

Elmaleh and Shankararaman [4] report on the impact of implementing a course, with 280 students, using FC. They observed that in comparison to a previous “traditional” running of the course, the FC increased pass rates in the final exam and also enhanced competency acquisition.

Wang et al. [22] present an experience report about using FC and PBL in a course with 132 students. They show that the adoption of FC and PBL brought significant enhancement to students’ performance in the final exam, compared to previous “traditional” running of the course. However, they also experienced inadequate preparation before class by some of the students and lack of enthusiasm in classroom interaction.

Mohamed [12] evaluates the extent to which FC, combined with pair programming, enhances students learning in a relatively small (90 students), mixed-ability CS1 course. The study showed that the use of FC increased the average class grade, pass rate, and course ratings compared to a previous corresponding non-flipped course.

Battestilli et al. [1] present preliminary results, from a course of 219 students, in which significant differences in students’ performance is identified based on how active they are in the online activities offered as part of the FC approach.

Sprint and Fox [18] present the implementation of a FC, gamified CS1 course, designed to motivate students to improve their study habits. The study showed that students in the FC course submitted programming assignments and online quizzes earlier and with fewer late submissions compared to students in a “traditional” course. Nevertheless, these improved study choices did not lead to higher final exam scores.

2.2 Online Courses

Students, teachers, and authorities had to adapt quickly to new pedagogical models and organisation of teaching delivery in 2020. Higher education institutions were informed by government mandates, executive orders, or recommended best practices to convert the modality of instruction to an online learning platform [5, 16].

Despite the fact that online courses have become increasingly popular at universities in the last few years [7, 19], the literature only contains (to our best of knowledge) a few recent papers about converting face-to-face classes to an online format for specific CS courses.

Irani and Denaro [7] describe their experience in creating an online course, with active learning strategies, in Discrete Mathematics and compare it to a corresponding “traditional” face-to-face course. They found no discernible difference in student performance between the two class versions.

Subramanian and Budhrani [20] redesigned an object-oriented systems face-to-face course as an online course. The content of the course was retained, while its structure was significantly modified to use a PBL approach. They found that various factors are critical for a successful online course, e.g. course structure, content scope, project design, assessment design, instructional resources and tools.

Related descriptions and studies can be found in other fields. Wang and Goryll [21] discuss the experience of changing a face-to-face 15-week lecture-lab Digital Design course to online format. Student retention and performance was similar between the online and the face-to-face course, and students reported overall satisfaction with the online course. Suggested improvements are related to development of course materials, better student support and integrating teamwork in the course.

Gottipatti and Shankaraman [5] present how a Master’s degree course in Text Analytic and Applications was rapidly moved over to an online format during the COVID-19 crisis. The lesson learnt was that changing a course from face-to-face to an online format is not an easy task, and that students’ feedback is valuable in the design process. The results suggested that it might be necessary to reduce the course content when using an online form, that the instructor needs time to learn how to master advanced features of the technology tools, and the student also need time to adjust to new methods of studying. Student evaluation showed that a large majority of the students were satisfied with the online course.

Roy and Covelli [16] describe moving courses in a liberal arts institution from a face-to-face format to an online format when half of the semester had passed. They found that the move, for both faculty and staff, was easier for those with prior experience with an online format and/or for those who felt comfortable with an online format. Moreover, a majority of the students expressed less interest than before in taking online classes.

Grimmer et al. [6] investigated the transition to online teaching (when the semester has already started) in an academic literacies course, and how to use the experience both for future online and face-to-face courses. They recognised online methods that could be used in face-to-face environment e.g., online classroom chat in lectures. The retention rates of their face-to-face students that were forced into the online learning environment was in line with the last five years in face-to-face learning in the course.

3 Background

The CS1 course at Reykjavik University is a large course. In fall 2020, 502 students, mainly from the departments of CS, Engineering, and Business, registered for the course. The students from CS take the course during their first semester, while the students from Engineering and Business pursue the course in their third semester. Our CS1 course thus “attracts a diverse crowd of students who bring mixed abilities and backgrounds to the classroom” [12].

The course does not assume any prior programming knowledge and uses Python to introduce fundamental CS1 programming concepts, e.g. variables, types, control structures, and functions, as well as built-in data structures like strings, lists, and dictionaries. The

2:4 Moving Classes Online

concept of a class is introduced and how it supports encapsulation and information hiding in the context of object-oriented programming. Students learn to use both an Integrated Development Environment (IDE) and command prompt mechanisms for the development and execution of programs.

In [10], we described our experience and the results of using a FC and a TBL approach in an introductory programming course during fall 2018. In fall 2019, we presented several improvements to the course [11], e.g. we reduced the pressure of submission of assignments during classes, emphasized textbook reading to a greater extent, showed short videos at the beginning of each class, and provided the students with several videos (for the student to watch at home) that demonstrated how to apply functional decomposition. A comparison of the results from student surveys given in both years showed that students were in general satisfied with the changes made and the performance on the exams was better in 2019 compared to 2018. The implementation in 2019 can be summarized as follows:

- One faculty member, the main instructor, was responsible for the overall organization of the course (syllabus, assessment, quizzes, projects, exams, etc.).
- The students were divided into several sections, with 50–70 students in most sections. Inside each section, the students were divided into teams of 5–6 students. Each section met two days a week in class, for four lecture hours each day. Each section/class had one teacher and one teaching assistant (TA) as facilitators and tutors. Whenever the students had a question they could ask the tutors for help.
- Students were supposed to come prepared for classes by reading specific chapters of the textbook [13] and, in some cases, by watching a video (made by the main instructor).
- At the beginning of each class, a 15–25 minutes video (made by the main instructor) was shown in each of the sections. Each video gave an overview of the concepts to be discussed/worked on in the class.
- A short individual quiz, containing ten multiple-choice questions, which were directly linked to the given textbook material and the video, was given in most of the classes after the video had been played. Thereafter, students discussed the same quiz in their teams, which turned in a single collective answer for each of the questions.
- For the remainder of the class, the students were given several short programming assignments to work on in the teams. Students were encouraged to work together on the solutions, but each student needed to submit his/her solution before the class finished. Students' solutions to each assignment were graded using automatic tests.
- In addition to the short programming assignments given during class, the students were given larger programming projects each week to be worked on at home, optionally in a group of two students.
- *Mimir Classroom*¹ was used for quizzes and programming assignments/projects. The quizzes and programming assignments in class were automatically graded by Mimir, whereas the weekly programming assignments were graded by a group of TAs. *Piazza*² was used as the question-answering platform, and *Canvas*³ as the Learning Management System.

The results from the student surveys, presented by the authors in [11] show that, overall, the students in the programming course liked to work in teams with fellow students, that the discussion with fellow students in class helped them to learn, and that they felt that communication with teachers in class helped them to study.

¹ <https://www.mimirhq.com/>

² <https://www.piazza.com/>

³ <https://www.instructure.com/canvas/>

4 Moving Classes Online

In light of the successful implementation described in Section 3, we intended to use the same format for the course in fall 2020. However, due to the COVID-19 pandemic, we needed to move the classes online. The main challenge we faced was how to exercise both TBL and FC when face-to-face classes were not allowed. We were indeed quite worried that applying these teaching methods would not turn out to be successful, when running the classes online.

In order to emulate a face-to-face class with student teams in each class, we used Zoom⁴ and its *breakout room* functionality. One recurring meeting (twice a week) was created in Zoom for each of the student sections. In addition, one (Live Q/A) thread was created in Piazza for each section for each meeting. Our course had nine sections, with about 55 students, on average, in each section.

Each typical synchronous meeting/class was run in Zoom in the following manner:

- At the beginning of each class, the teacher and the TA “met” all students in the main room. The teacher started the class with an overview of the schedule for the day and then played a pre-recorded, short video (see Section 3), in which an overview was given of the concepts to be discussed/worked on in the class.
- After the video had been played, students were able to ask questions before the quiz started in Mimir. In addition to the individual quizzes, in 2018 and 2019 we had used team quizzes as well, implemented with Immediate Feedback Assessment Technique (IF-AT) scratch cards [10, 11]. Because these cards are “physical” cards, we were not able to include the team quizzes in our online classes.
- Once the quiz was finished, the programming assignments were opened in Mimir, to be worked on in the teams. Each team was given a separate breakout room in Zoom. At the beginning of the semester, students were randomly allocated into teams, but after about three weeks the students were allowed to form their own pre-defined teams. If a pre-defined team consisted of less than five students, the teacher moved students, not belonging to any team, into the pre-defined ones by using the corresponding functionality in Zoom. When students in a breakout room needed help, they posted a help request on the corresponding Piazza thread, monitored by the teacher and the TA.
- The teams were not given any special instructions on how to work collaboratively on the programming assignments in class. In some cases, each team member wrote his/her own code and mainly shared ideas in the breakout rooms. In other cases, the team members applied pair/tri programming. As mentioned in Section 3, before the end of the class each team member needed to submit his/her solution in Mimir. Visual Studio Code⁵ was used as the main programming development environment.

Apart from the use of Zoom, the organization and implementation of our course closely follows the one presented in Section 3.

However, there is one other difference. In [10], we specifically noted that there were two “problems” with students with previous programming knowledge, who are, in most cases, male students. First, some of the female students felt intimidated by these male students in the team work. Second, many of the students with previous programming knowledge felt bored during the first weeks of the course, because the material was too elementary for them.

⁴ Zoom is a video-conferencing solution which supports high quality point-to-point and multi-party video conferencing, content sharing, and group and individual chat [14].

⁵ <https://code.visualstudio.com/>

■ **Table 1** Student assessment.

Item	Weight
Quizzes in class	10%
Programming assignments in class	10%
Weekly programming projects	20%
Two midterms	0–20%
Final exam	40–60%

■ **Table 2** Answers to the question “Are you generally pleased or displeased with the course?”.

Rating	Answer	Count	Ratio
5	Very pleased	116	36.3%
4	Rather pleased	121	37.8%
3	Moderate	53	16.6%
2	Rather displeased	22	6.9%
1	Very displeased	8	2.5%

We decided to implement our own suggestion put forth in [10], i.e. by making a special section for students with previous programming knowledge and present additional, more challenging, programming assignments to them. The establishment of this special section was announced a week before the course started and participation in the section was optional. About 60 students, of the 502 starting the course, signed up for this special section.

As we pointed out in [10], a FC/TBL version of a large CS1 course demands considerable effort and manpower. In our course, in addition to the main instructor, we had 16 teachers and TAs in the nine sections, eight TAs graded the weekly homework assignments, two TAs took care of a special helping session once a week, and two TAs were hired to specifically help the main instructor to answer questions on Piazza. Thus, about 30 persons contributed to the running of the course in some way or another.

The students’ assessment consists of the five items shown in Table 1. To obtain full points for the programming assignments in class, the students needed only to pass 50% of the automatic tests, on average, during each class. This rule was introduced in 2019 to reduce the pressure many of the students felt when needing to submit solutions to all assignments during each class [10]. Each of the two midterm exams weighted 10%. However, if the student obtained a higher grade on the final exam than in one or both of the midterm exams, the weight of the final exam increased to either 50% or 60% and the corresponding midterm weight dropped to 0%.

5 Surveys

In this section, we present and discuss the results of two surveys carried out among students registered in the course.

5.1 First survey

The first survey (student evaluation) was administered by the Office of Teaching Affairs and was given to students in the fifth week of the course. At that time, 485 students were still registered in the course, of which 320 students (66%) participated in the survey. Students were asked to answer a single question rated on a five point Likert scale. The question and the results are presented in Table 2.

According to the results, 74.1% of the students were pleased (either very pleased or rather pleased) with the course, and only 9.4% displeased (rather displeased or very displeased). The weighed average is 3.98. These results are very similar to the ones obtained for the same question in the course in fall 2019, where 76% of the students were pleased with the course, 7% displeased, and for which the weighted average was 4.0 [11]. According to this comparison, moving the classes online did not affect students' satisfaction in the first weeks of the course.

5.2 Second survey

In order to evaluate students' attitudes, experience and learning, we constructed a detailed survey, consisting of 28 questions, which students answered during weeks 9–10 of the course. At that time, 473 students were still registered, and 305 students (64.5%) provided answers to the questions. We used the questions from the 2019 course [11], but, in addition, we added a few questions connected to the fact that our classes had been moved online.

We divide the questions into four main categories: *Course organization and teaching* (nine questions), *Study materials and midterms* (six questions), *Cooperation and communication* (four questions), and *Use of systems* (four questions)⁶. Where applicable, in the tables we provide the results from the same questions in fall 2019, in parentheses, i.e. “(19)”. In Tables 3–6, the columns “Totally agree”, “Agree”, “Neutral”, “Disagree”, and “Totally disagree” correspond to points 5–1, respectively, on the Likert scale.

By considering the results presented in Table 3, we see that the responses are very similar between 2020 and 2019 (a t-test between the mean scores of the two years revealed no significant difference in any of the nine questions). Students are generally satisfied with the organization and the learning experience in the course: 66% of the students agree (sum of columns “Totally agree” and “Agree”) that the organization of the course is good (question one), and 69% agree that the course is overall a good learning experience (question two). The results from the first survey, presented in Table 2, show that about 74% of the students were pleased with the course after five weeks. It is understandable that satisfaction drops a bit when the course progresses, because the material gradually becomes harder. When considering the results from questions one and two it can be deduced that by moving the classes online has not reduced students' satisfaction between the years 2019 and 2020.

It is interesting that, despite the general satisfaction among students, 42% agree that the course lacks traditional lectures (question four) and that only about half of the students agree that using the FC is suitable for the course (question five). On the other hand, note that only 14% disagree that using the FC is suitable.

According to the answers to question nine, 54% of the students agree that moving the classes online is suitable, whereas 31% disagree. Due to the general satisfaction with the organization and the learning experience, one might be inclined to keep this online format of the classes in the future, but one may have to take into account that about one-third of the students disagree of the suitability of the online move.

According to Table 4, 55% of the students in 2020, compared to 48% in 2019, feel that the textbook helped in their studies (question 10). At the beginning of our course, we specifically emphasized the importance of the textbook and that students should come prepared for classes by reading given chapters of the text. Nevertheless, according to question 11, only 47% of the students usually read the textbook before class and this figure is a bit lower compared

⁶ In the accompanying tables, we skip five questions that mainly concern students' background.

■ **Table 3** Course organization and teaching.

Question	Totally agree 20 (19)	Agree 20 (19)	Neutral 20 (19)	Disagree 20 (19)	Totally disagree 20 (19)
1. The organization of the course is good	26% (20%)	40% (42%)	22% (23%)	7% (12%)	5% (3%)
2. This course is overall a good learning experience	36% (33%)	33% (34%)	17% (18%)	10% (9%)	5% (6%)
3. The classes each week are useful to me	28% (29%)	30% (29%)	23% (20%)	14% (16%)	5% (6%)
4. The course lacks traditional lectures	24% (25%)	18% (17%)	21% (19%)	17% (19%)	20% (20%)
5. I feel that the flipped classroom is a suitable approach in this course	24% (24%)	24% (29%)	38% (24%)	7% (12%)	7% (10%)
6. I like the assessment of the programming assignments in class	49% (43%)	30% (29%)	13% (16%)	5% (6%)	3% (6%)
7. The programming assignments in class are in accordance with the teaching material	35% (37%)	31% (32%)	17% (14%)	9% (10%)	8% (7%)
8. The weekly progr. projects are in accordance with the teaching material	30% (27%)	28% (25%)	20% (20%)	13% (16%)	9% (12%)
9. I feel that moving the classes online is suitable in this course	33%	21%	16%	14%	17%

to 2019. Note that a substantially higher ratio of students watch the videos before/after class (question 13), compared to the ratio of students reading the textbook. For Table 4, a t-test between the mean scores of the two years revealed no significant difference.

The fact that less than 50% of the students read the textbook before coming to classes is worrying, but not surprising. For several years, we have noticed a gradual decline in textbook reading by our CS students. According to Brown et al. [3], “one issue that plagues millennials is the lack of focused, in-depth reading to achieve understanding”, and “one of the biggest challenges instructors have dealt with for many years is getting students to read their required textbooks”.

Table 5 (question 16) shows that a lower ratio of students in 2020 (63%) compared to 2019 (73%) agree that discussing with fellow students helped their studies (a t-test between the mean scores of the two years only revealed a significant difference for this question (t-value 3.11, $p < 0.01$)). This may indicate that the communication between students in the Zoom breakout rooms is not as straight-forward as it is when communicating in person. The teachers noted that in some teams the communication between students in the breakout rooms was generally active and that they worked collaboratively, while in other teams it seemed that the students worked rather as individuals. On the other hand, the responses to question 19 show that a higher ratio of students in 2020 (62%) compared to 2019 (55%)

■ **Table 4** Study material and midterms.

Question	Totally agree 20 (19)	Agree 20 (19)	Neutral 20 (19)	Disagree 20 (19)	Totally disagree 20 (19)
10. The textbook of the course helped me in my studies	22% (25%)	23% (23%)	25% (28%)	17% (16%)	12% (8%)
11. I usually read the book before the class	26% (33%)	21% (19%)	18% (18%)	13% (11%)	21% (19%)
12. The videos of the course helped me in my studies	23% (24%)	33% (28%)	26% (27%)	12% (16%)	7% (5%)
13. I usually watch the videos given before/after the class	42% (42%)	25% (27%)	20% (16%)	6% (10%)	7% (5%)
14. I like taking a quiz at the beginning of class	18% (19%)	30% (31%)	28% (23%)	13% (14%)	12% (12%)
15. I like the arrangements of the midterm exams	53% (57%)	31% (31%)	11% (8%)	3% (0%)	1% (3%)

feel that communication with the instructors in class helped their studies. The difference between the mean scores of the two years for this question is not statistically significant, but it may be the case that it is easier for a student team to communicate with a teacher inside a breakout room as opposed to the team sharing a large physical classroom with all other students (as in 2019).

Table 6 shows that students are generally happy with the support systems used in the course. However, there is a large reduction of satisfaction in Piazza usage (question 21) in 2020 (46%) compared to 2019 (63%). This is difficult to explain, but we conjecture that students did not like to ask teachers to “visit” them in Zoom breakout rooms by posting such a help message in Piazza, where it can be seen by everyone. Unfortunately, in the Zoom version we used, it was not possible to post a message to the host or co-host (teacher or TA) from within a breakout room. This is the reason why we used Piazza for this purpose. For Table 6, a t-test between the mean scores of the two years only revealed a significant difference for question 21 (t-value 4.11, $p < 0.001$).

6 Exams

In this section, we present the implementation of the two midterm exams, the final exam, and the retake exam given in the course along with exam results. All four exams were set up in Mimir, and students were able to receive assistance by using Piazza and Zoom. If a student needed assistance during an exam, he/she posted a request in a given Piazza thread and, consequently, the teacher or the TA moved the student into a breakout room in Zoom.

The exams were “open book”, in the sense that students were allowed to use the textbook, slides, notes, and solutions to assignments in the exam. Students were, however, neither allowed to use web search nor any kind of communication software during the exam. Grades are given on a 0–10 scale, and a grade below 5 is a failing grade.

2:10 Moving Classes Online

■ **Table 5** Cooperation and communication.

Question	Totally agree 20 (19)	Agree 20 (19)	Neutral 20 (19)	Disagree 20 (19)	Totally disagree 20 (19)
16. Discussing with fellow students in class helped me in my studies	35% (44%)	28% (29%)	19% (16%)	9% (6%)	9% (4%)
17. Discussing with fellow students outside class helped me in my studies	35% (39%)	31% (31%)	22% (20%)	7% (4%)	6% (6%)
18. I like to work in a group with fellow students	35% (38%)	27% (29%)	21% (19%)	10% (8%)	7% (5%)
19. Communication with instructors in class helped me in my studies	28% (32%)	34% (23%)	21% (26%)	12% (12%)	5% (8%)

■ **Table 6** Use of systems.

Question	Totally agree 20 (19)	Agree 20 (19)	Neutral 20 (19)	Disagree 20 (19)	Totally disagree 20 (19)
20. I like to use Canvas	37% (39%)	33% (29%)	26% (26%)	2% (4%)	1% (2%)
21. I like to use Piazza	23% (34%)	23% (29%)	26% (24%)	16% (5%)	12% (8%)
22. I like to use Mimir	45% (52%)	36% (30%)	10% (13%)	4% (3%)	5% (2%)
23. I like to use Zoom	31%	31%	19%	11%	7%

The plagiarism software *Moss*⁷ was run on student solutions. We found some plagiarism cases in the midterm exams. These cases were reported (resulting in the grade 0) and, consequently, the seriousness of plagiarism was discussed with the whole student body on Piazza. As a result, we did not find any obvious cases of plagiarism in the final exam.

We used the timing and the material for the exams described in [10]:

The first midterm exam was given in the fourth week of the course. The material for the exam were basic programming concepts like variables, types, operators, assignment statements, expressions, if-statements, and loops. The second midterm exam was given in the eighth week of the course. In addition to the material covered in the first exam, the second one included the following concepts: functions and top-down refinement, scope, file I/O, exception handling, lists and tuples. At the time of the final exam, the following concepts had been added: dictionaries, sets, (large) program development, and classes.

The duration of the first midterm exam, the second midterm exam, and the final/retake exam, was 2 hours, 2.5 hours, and 3 hours, respectively.

Table 7 shows the results from the four exams – the numbers in parenthesis show the results from the 2019 course⁸. The number of students still registered in the course at the time of the four exams were 485, 474, 463 and 463, respectively. According to our experience,

⁷ <https://theory.stanford.edu/~aiken/moss/>

⁸ The student body and the difficulty of the exams in the courses in 2019 and 2020 are very similar, which justifies making a direct comparison.

■ **Table 7** Exam results.

Exam	Students 2020	Average grade 2020 (2019)	Failure rate 2020 (2019)
Midterm 1	462, 95.3%	8.9 (8.5)	3.5% (8.1%)
Midterm 2	433, 91.3%	5.8 (6.0)	36.7% (39.6%)
Final	401, 86.6%	6.0 (5.9)	35.4% (33.9%)
Retake	125, 27.0%	4.9 (4.0)	39.2% (52.7%)

the participation in the second midterm exam is a good indicator of the dropout rate in the course. In 2019, 82.8% of the registered students (at the start of the course) showed up in the second midterm, whereas the corresponding ratio in 2020 was 86.3%.

It is noteworthy how much the average grade decreases in midterm 2 in comparison to midterm 1, and, consequently, how much the failure rate increases. This is consistent with the exam results for the 2019 course. The reason is that the material covered on the first midterm exam is relatively easy for most students, whereas the second midterm exam covers more complex concepts.

As mentioned in Section 4, students with previous programming experience were given the opportunity to take part in a special section intended for this group of students. The average grade on the final exam for students in this special section was 8.4, compared to 5.7 for the students in the other sections. This difference in final exam performance in CS1 between students with and without previous programming experience is even greater than, for example, the results presented in [24].

The grades in the two midterm exams and the final exam in 2020 are very similar to the corresponding grades from the 2019 course. We thus deduce that moving our classes online did not significantly affect students' performance.

Of the 502 students that were registered at the start of the course, 353 students (70.3%) passed the course (either the final exam or the retake exam). The corresponding failure rate of 29.7% is a bit lower than the mean worldwide failure rate of 32.3% presented in [23].

7 Conclusion

In this paper, we presented an experience report on moving the classes in a large CS1 course, emphasising FC and TBL, to an online format in 2020. The motivation for the move was the COVID-19 pandemic. A priori, we were worried that moving the classes online would make it difficult to successfully use the FC and TBL teaching and collaborative approaches in our programming course. However, our experience shows that conducting these methods in online classes did not pose any special problems in comparison to face-to-face classes.

We presented the results of two surveys and four exams. According to the surveys, students were generally satisfied with the organization of the course and the learning experience. A comparison of the results to the course from the previous year shows that moving the course online did neither have negative effects on students' attitudes nor on students' grades.

When the on-going pandemic is over, we need to make a decision on whether to move back to face-to-face classes in our CS1 course. It might be an option to continue giving the classes online in the future. One argument against online classes is that, according to our results, a lower ratio of students in these classes feel that discussing with fellow students helped their studies, compared to the similar face-to-face class course from the previous year. On the other hand, our results also show that a higher ratio of students in the 2020 course, compared to the course in 2019, feel that communication with the teachers in class helped their studies.

Finally, the best option may be to allow students to choose the class format that best fits their individual needs, i.e. allowing students to choose either online classes or face-to-face classes.

References

- 1 Lina Battestilli, Ignacio X. Domínguez, and Maanasa Thyagarajan. Toward Finding Online Activity Patterns in a Flipped Programming Course. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, page 1345, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3328778.3372626.
- 2 Brett A. Becker and Keith Quille. 50 Years of CS1 at SIGCSE: A Review of the Evolution of Introductory Programming Education Research. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, page 338–344, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3287324.3287432.
- 3 Charles A. Brown, Kreag Danvers, and David T. Doran. Student Perceptions on Using Guided Reading Questions to Motivate Student Reading in the Flipped Classroom. *Accounting Education*, 25(3):256–271, 2016. doi:10.1080/09639284.2016.1165124.
- 4 Joelle Elmaleh and Venky Shankaraman. Improving student learning in an introductory programming course using flipped classroom and competency framework. In *2017 IEEE Global Engineering Education Conference (EDUCON)*, pages 49–55, 2017. doi:10.1109/EDUCON.2017.7942823.
- 5 Swapna Gottipatti and Venky Shankaraman. Rapid Transition of a Technical Course from Face-to-Face to Online. *Communications of the Association for Information Systems*, pages 1–8, 2021. URL: https://ink.library.smu.edu.sg/sis_research/5401.
- 6 Raelke Grimmer, Andrew Pollard, and Nicola Rolls. COVID-19 induced change in higher education: Reflections on rapidly transitioning a first-year undergraduate academic literacies unit from face-to-face to online. *Journal of Academic Language and Learning*, 14(2):95–105, 2020. URL: <https://journal.aall.org.au/index.php/jall/article/view/695>.
- 7 Sandy Irani and Kameryn Denaro. Incorporating Active Learning Strategies and Instructor Presence into an Online Discrete Mathematics Class. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, page 1186–1192, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3328778.3366904.
- 8 Krisztina V. Jakobsen and Megan Knetemann. Putting Structure to Flipped Classrooms Using Team-Based Learning. *International Journal of Teaching and Learning in Higher Education*, 29(1):175–185, 2017.
- 9 Patricia Lasserre. Adaptation of Team-based Learning on a First Term Programming Class. In *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITiCSE '09, page 186–190, New York, NY, USA, 2009. Association for Computing Machinery. doi:10.1145/1562877.1562937.
- 10 Hrafn Loftsson and Ásrún Matthíasdóttir. Using Flipped Classroom and Team-Based Learning in a First-Semester Programming Course: An Experience Report. In *Proceedings of IEEE International Conference on Teaching, Assessment, and Learning for Engineering*, TALE '19. IEEE, 2019. doi:10.1109/TALE48000.2019.9225985.
- 11 Ásrún Matthíasdóttir and Hrafn Loftsson. Improving the Implementation of a First-Semester Programming Course. In *Proceedings of the 16th International CDIO Conference*, Gothenburg, Sweden, 2020. CDIO. URL: <http://www.cdio.org/content/improving-implementation-first-semester-programming-course>.
- 12 Abdallah Mohamed. Evaluating the Effectiveness of Flipped Teaching in a Mixed-Ability CS1 Course. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '20, page 452–458, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3341525.3387395.

- 13 William F. Punch and Richard Enbody. *The Practice of Computing Using Python*. Pearson Education, New York, NY, 3rd. edition, 2017.
- 14 Wainhouse Research. Evaluation of the Zoom Cloud Video Conference Service, 2015. URL: <https://zoom.us/whitepaper?case=wainhouse>.
- 15 Anthony Robins, Janet Rountree, and Nathan Rountree. Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, 13(2):137–172, 2003.
- 16 Sudipta Roy and Bonnie Covelli. COVID-19 Induced Transition from Classroom to Online Mid Semester: Case Study on Faculty and Students’ Preferences and Opinions. *Higher Learning Research Communications*, 11:10–32, 2020. doi:10.18870/hlrc.v11i0.1197.
- 17 Manoj D. Souza and Paul Rodriques. Investigating the Effectiveness of the Flipped Classroom in an Introductory Programming Course. *The New Educational Review*, 40(1), 2015. doi:10.15804/tner.2015.40.2.11.
- 18 Gina Sprint and Erik Fox. Improving Student Study Choices in CS1 with Gamification and Flipped Classrooms. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE ’20*, page 773–779, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3328778.3366888.
- 19 Christian Stöhr, Christophe Demazière, and Tom Adawi. The polarizing effect of the online flipped classroom. *Computer Education*, 147:103789, 2020. doi:10.1016/j.compedu.2019.103789.
- 20 Kalpathi Subramanian and Kiran Budhrani. Influence of Course Design on Student Engagement and Motivation in an Online Course. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE ’20*, page 303–308, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3328778.3366828.
- 21 Chao Wang and Michael Goryll. Design and Implementation of an Online Digital Design Course. In *123rd ASEE Annual Conference and Exposition*, volume 2016-June. American Society for Engineering Education, 2016. URL: <https://asu.pure.elsevier.com/en/publications/design-and-implementation-of-an-online-digital-design-course>.
- 22 Gang Wang, Hong Zhao, Yun Guo, and Min Li. Integration of Flipped Classroom and Problem Based Learning Model and its Implementation in University Programming Course. In *2019 14th International Conference on Computer Science Education (ICCSE)*, pages 606–610, 2019. doi:10.1109/ICCSE.2019.8845525.
- 23 Christopher Watson and Frederick W.B. Li. Failure Rates in Introductory Programming Revisited. In *Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education, ITiCSE ’14*, page 39–44, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2591708.2591749.
- 24 Chris Wilcox and Albert Lionelle. Quantifying the Benefits of Prior Programming Experience in an Introductory Computer Science Course. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE ’18*, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3159450.3159480.



Programmers' Affinity to Languages

Alvaro Costa Neto  

Federal Institute of Education, Science and Technology of São Paulo, Barretos, Brazil

Cristiana Araújo   

Centro ALGORITMI, Departamento de Informática,
University of Minho, Campus Gualtar, Braga, Portugal

Maria João Varanda Pereira   

Research Centre in Digitalization and Intelligent Robotics,
Polytechnic Institute of Bragança, Portugal

Pedro Rangel Henriques   

Centro ALGORITMI, Departamento de Informática,
University of Minho, Campus Gualtar, Braga, Portugal

Abstract

Students face several challenges when learning computer programming languages, a central topic to acquire programming skills. While those challenges that present a predominantly technical nature have been intensely studied by researchers along the years, the ones that are concerned with qualitative, and personal aspects have not. Affinity to a programming language is one of the many personal factors that may contribute to surpass these qualitative aspects that describe the difficulties that students face. From this point-of-view, this paper presents a proposal for treating and studying programmers' affinity to programming languages as an important factor for learning computer programming. It also reports a preliminary questionnaire conducted on a master's degree class at Universidade do Minho that showed that affinity may have a broader relation to learning computer programming than anticipated. Finally, a set of relevant questions are stated to compose a future inquiry aimed at deepening the knowledge on the affinity between programmers and languages, paving the way for following research.

2012 ACM Subject Classification Applied computing → Computer-assisted instruction

Keywords and phrases Computer programming, Programming Languages, Affinity, Education, Learning

Digital Object Identifier 10.4230/OASICS.ICPEC.2021.3

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

Learning a complex topic such as computer programming is an intricate process, influenced by several factors of varied nature. Research in this field dates back several decades [5], and has kept constant academic interest throughout the years [6, 15, 12, 3], usually through a technical approach – development and evaluation of tools that are used to teach and learn. While of great value, these are not the only problems that students face when learning. In fact, when observed through a pedagogical lenses, many other characteristics of different nature also seem to explain how knowledge is constructed, allowing food for thought on how to facilitate this process in a more personal perspective. The widely known researcher Jean Piaget stated that context, especially in its social branch, has a decisive impact on learning by providing the necessary interactions that adapt and organize the mental schemes [11]. Lev Vygotsky, another very influential pedagogy researcher, also highlighted the crucial importance that contextual factors play in the construction of knowledge [14], while positioning language



© Alvaro Costa Neto, Cristiana Araújo, Maria João Varanda Pereira, and Pedro Rangel Henriques; licensed under Creative Commons License CC-BY 4.0

Second International Computer Programming Education Conference (ICPEC 2021).

Editors: Pedro Rangel Henriques, Filipe Portela, Ricardo Queirós, and Alberto Simões; Article No. 3; pp. 3:1–3:7

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

3:2 Programmers' Affinity to Languages

as the main mediator for learning. Paulo Freire went one step further by supporting the idea that teaching is a force of change to socioeconomic contexts, which is only possible by understanding and aggregating the students' backgrounds into the process [7]. In a different recent project [4], focused in adults' professional reconversion, we also realized the relevance of background and know-how in their learning process, including the impact on the choice of the best learning resources. Based on these authors' findings, it's reasonably safe to assume that personal factors are inevitably involved in the process of knowledge construction, and therefore should be taken into account when teaching.

As one of these personal factors, *affinity* may sustain an important relation to the learning process, even on very technical subjects such as computer programming, and it deserves, for sure, to be further analysed. While there have been studies on personal factors being influential to the education process as a whole, affinity's influence is still largely unexplored when it comes to computer programming learning. Affinity's qualitative nature and the fact that it may be directly related to technical aspects of the programming languages present a dual, albeit interesting conundrum to investigate. As such, many research questions emerge when affinity is considered an important feature of the learning process:

- Which of the language's characteristics stimulate students' subconsciousness, subtly conducting them to like or dislike certain programming languages?
- Do these characteristics influence students positively or negatively?
- Is the inverse actually true, meaning that the first programming language learnt strongly influences the consequent student's affinity with that particular language?

Finding answers to these questions is the main goal of the study discussed in this paper, aiming to instigate further research on this subject.

This paper is structured in five sections: after the introduction, section 2 discusses which factors may influence the learning process and how they may be categorized. The third section presents affinity as an important and challenging topic for discussion when considering the learning of computer programming. Moreover, in this section it is also presented the initial investigation that underlies the work here reported. Based on the previously presented results, section 4 raises relevant questions that must be answered in order to better understand the role of affinity in learning, and in the development of programming skills. Finally, section 5 concludes this paper presenting a proposal of further investigation that will be conducted.

2 Influences on Learning Computer Programming

Many factors influence directly or indirectly the learning process. Whenever possible, it is good practice to take into account these influences in order to improve the student's experience. Three main categories may be established: teaching methods, programming languages characteristics and personal background.

The *teaching methods* category contains the array of factors based on the application of pedagogical methods, either formally stated or experience based. Tools and other mediating resources are also included in this category, such as proposed by [1, 2]. The most common factors include:

- Quality of the communication process;
- Media diversification;
- Auxiliary tools;
- Clarity of the explanation *etc.*

Factors that are directly connected to the underlying principal and design of the programming language are its formal definition, paradigm, syntax, semantics, and quality [10]. Also important are intrinsic characteristics such as:

- Expressiveness;
- Viscosity;
- Verbosity
- Readability;
- Consistence;
- Error-proneness;
- Abstraction level *etc.*

Historical, contextual, and subconscious characteristics form the *personal background* category. Examples of such factors include:

- Socioeconomic context;
- Previous contact with formal logic and computers;
- Experience with the English language [9];
- Psychological issues such as being away from family for the first time [8];
- Affinity with specific languages and tools;
- Lack of interpersonal skills *etc.*

Figuring in as a personal and complex factor, *affinity* may be established by the students to specific programming languages. On one hand, considering affinity may lead to new paths to motivate students improving their learning process. On the other hand, affinity might also be seen as a result of the students' experiences, guided by their courses' curricula and the initial languages he or she has learnt.

3 Affinity to Programming Languages

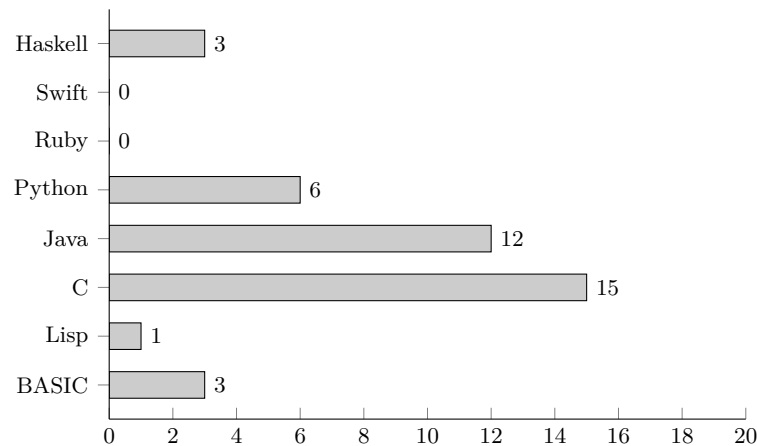
The previous statements are supported by a preliminary indication observed by asking twenty three students of a Masters' degree class in Computer Engineering at Universidade do Minho to answer a few questions about programming languages and their learning experiences. After a lecture on teaching computer programming, in which many of these topics were discussed, some snippets of code were shown in different languages – BASIC, Lisp, C, Java, Python, Ruby, Swift, and Haskell – and students were asked to point out which languages they like most, being C, Java and Haskell the three highest ranked. Then, a questionnaire was applied containing these questions:

1. In a range of very low to very high, how important is the language choice for learning computer programming? Justify your answer;
2. Out of those factors previously mentioned in Section 2 and presented in the lecture, which ones are most relevant, and influential to learning computer programming?
3. Which languages would you choose to teach computer programming: BASIC, Lisp, C, Java, Python, Ruby, Swift, or some other language? Justify your answer. This was a multiple choice question, allowing students to choose more than one language.

If popularity and market share were the main influences driving the students' choices, it would be safe to assume that Python would be the highest ranked option, since it's currently a very popular programming language [13], with many popular applications, such as artificial intelligence and numerical computing. Nonetheless, C and Java overthrew Python as shown

3:4 Programmers' Affinity to Languages

in Figure 1. Haskell also had a perceptive presence in the results, which is also unexpected from the perspectives of current market share and ease of use. These surprises become understandable if the affinity to the language was actually constructed as a *consequence of their learning experiences*, since these students had been formally taught Haskell, C and Java as their initial programming languages. While the current attractiveness of Python can't be discarded – it ranked third, after all – these results show that programming language affinity is more subtle and complex than expected, and could ultimately influence other activities beyond learning.



■ **Figure 1** Language choices by students when asked which languages they prefer.

These unexpected findings imply that many questions should arise from more exploratory observation of how students learn computer programming, creating opportunities for deeper investigation on affinity with programming languages and its relation to the learning process.

4 A Survey to Understand Affinity

Learning how to correctly program a computer involves not only understanding how to type commands that are syntactically correct. It also involves understanding problems, abstracting concepts and finding recurrent patterns, delineating their boundaries, designing algorithms, and translating viable solutions to source code. In this process, the programming language acts simultaneously as a tool and as an interface of communication, interfering with both learning and problem solving. Having a close understanding of its rules is mandatory for obtaining good results, but solving a problem using a language enacts a more complex intellectual mechanism, specially when learning.

It becomes clear that the mental schema that must be constructed in order to program an application involves many aspects of the human mind: memory, perceptiveness, and motivation, to name a few. As is safe to assume that affinity is an important part of motivation, its proper usage can be a valuable aid for education professionals. Several interesting questions arise to help understanding affinity and the role it might play in learning computer programming:

- How does affinity to a programming language come to be?
- Which factors affect its growth, either hampering or stimulating it?
- Is this affinity a support to or a consequence of the learning process?
- How do teaching methodologies come into play to stimulate it?
- Which personal factors affect affinity growth?

These questions, while of central importance, are too broad to directly compose a survey. They serve to structure the objectives by instigating the search for their answers. In order to obtain tangible results though, it's necessary to delve deeper into the problem of affinity, and pinpoint specific characteristics through a more palpable inquiring. This process, in and of itself, proposes an interesting challenge: how to obtain meaningful answers for a learning factor that may present itself simultaneously technical and personal, such as affinity?

This duality of nature may require an equivalent dual approach. While it's important to directly ask questions such as "Is this language appealing to you? Why?", or "Rank your favorite languages" to paint a personal picture of the respondent's background, this strategy may not work if applied to more technical aspects of the programming languages. In this case, direct questions may lead to colored answers, either by previous personal opinions, or by misinterpretation of what is being asked. A dual approach would apply indirect observation of affinity on technical aspects – comparing snippets of source code, would be an example – while still relating these aspects to the respondent's background, via more open, direct, and personal questions. This approach seems to be a more encompassing way to obtain meaningful answers, at least for the previously stated questions.

According to the previous discussion, we propose a survey based on the following structure:

1. A collection of personal questions in order to capture a picture of the respondents' contexts and backgrounds. It would also be interesting to ask directly which languages are their favorites and why, for later comparison with the answers given to the other questions;
2. Direct questions asking the respondents to analyze snippets of source code in various programming languages. These questions should be as direct and as clear as possible to avoid misinterpretations, and to accommodate the fact that not all respondents may be familiar with all presented languages. Several different languages should be applied – randomly, if possible – in order to assess their capacities of understanding source code, while simultaneously identifying direct and indirect characteristics that emerge as possible influential factors for the respondents' affinity to the languages that were chosen as their favorites;
3. A series of indirect questions, also applying several different programming languages and snippets of source code. These questions must be composed in such a way to detect indirectly which characteristics of the presented code are influencing the respondents' affinity to the presented languages, given their backgrounds and current knowledge;
4. Some direct questions inquiring what is his or her perception of the languages and snippets previously presented. These questions would serve as a leveling reference for the respondents' personal – and biased – view of the programming languages and what he or she thinks is more influential to create affinity. The answers for these questions are certain to form interesting comparisons with the ones given to the questions proposed on item 1.

This structure will support the construction of the survey. Implementation details, such as the order of the questions, which programming languages should be used, and complexity of the source code snippets, are yet to be determined. It would be ideal to apply dynamically constructed surveys, that would serve two main purposes:

- To provide specific inquiries for respondents that have different backgrounds, in order to minimize personal bias on more direct and technical questions, and;
- To design some mechanism of comparison between respondents that have experience with certain languages and those who have not.

The technical resources to implement these dynamic surveys are yet to be investigated and developed, if required. While having education purposes in mind, the survey will be freely available, and account for answers given by people other than students and teachers. Computer programming professionals, mathematicians, applied computing users, to list a few, may have very interesting, and important, views on affinity to their favorite programming languages.

Finally, the answers will be statistically correlated to the respondents' background and technical knowledge in order to obtain a better understanding of how and why affinity to a programming language is established. The final goal is to identify which characteristics of the current, and past programming languages are most related to the rise and fall of programmers' affinity. This information may be a valuable resource to aid educators to better choose – or when possible, allow the choice – of the programming language used in their classes.

5 Conclusion

The problem of more efficiently teaching computer programming still poses relevant, and appealing challenges. While many studies have shown diverse characteristics as influential to the learning experience, affinity to a programming language is still a largely unknown aspect.

Preliminary results point toward further investigation, as the conducted studies show that there are at least two – probably complementary – perspectives on affinity: it may be viewed both as a propelling factor to knowledge construction, and as a result of experience while learning computer programming. The former indicates that educators must be aware of the relationship that students develop with the programming languages they taught, specially in their initial stages of education. This observation, and consequent adaptation of the teaching process should improve students' learning experiences. The later – considering affinity as a product of the learning experience itself – may become a valuable indication to better understand how the programming languages taught in formal education influence the industry. On one hand, this affinity may induce future professionals – and consequently, their workplaces – to prefer and use the languages they were taught. On the other hand, it may create a resistance in the workforce to adopt current established languages. Nonetheless, in both perspectives affinity should be considered an important factor of the teaching-learning process, and consequently be taken into account when deciding which languages students should learn, work and improve.

In the near future deeper studies shall be conducted, investigating which specific characteristics contribute to language affinity, both technical, and personal in nature.

References

- 1 M. V. P. Almeida, L. M. Alves, M. J. V. Pereira, and G. A. R. Barbosa. EasyCoding - Methodology to Support Programming Learning. In Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões, editors, *First International Computer Programming Education Conference (ICPEC 2020)*, volume 81 of *OpenAccess Series in Informatics (OASISs)*, pages 1:1–1:8, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/OASISs.ICPEC.2020.1.
- 2 M.V.P. Almeida. Easycoding: Methodology to support programming learning. Master's thesis, Instituto Politécnico de Bragança, 2020.
- 3 A.G. Applin. Second language acquisition and cs1. *SIGCSE Bull.*, 33(1):174–178, February 2001. doi:10.1145/366413.364579.

- 4 D.R. Barbosa. CnE-Ar: Teaching of Computational Thinking to Adults in Reconversion. Master's thesis, Minho University, Braga, Portugal, April 2021. MSc dissertation (to be discussed and published).
- 5 R.R. Fenichel, J. Weizenbaum, and J.C. Yochelson. A program to teach programming. *Communications of the ACM*, 13(3):141–146, March 1970. doi:10.1145/362052.362053.
- 6 J. Figueiredo and F.J. García-Peñalvo. Building skills in introductory programming. In *Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality*, TEEM'18, page 46–50, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3284179.3284190.
- 7 P. Freire. *Pedagogia da Autonomia: Saberes necessários à prática educativa*. Paz e Terra, 2011.
- 8 A. Gomes and A.J. Mendes. Learning to program: difficulties and solutions. In *Proceedings of the 2007 ICEE International Conference on Engineering and Education*, ICEE '07. International Network on Engineering Education and Research, 2007.
- 9 P.J. Guo. Non-native english speakers learning computer programming: Barriers, desires, and design opportunities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–14, New York, NY, USA, 2018. Association for Computing Machinery.
- 10 P.R. Henriques. *Brincando às Linguagens com Rigor: Engenharia gramatical*. PhD thesis, Universidade do Minho, 2013.
- 11 J. Piaget, M. Piercy, and D.E. Berlyne. *The Psychology of Intelligence*. Routledge classics. Routledge, 2001.
- 12 S.A. Robertson and M.P. Lee. The application of second natural language acquisition pedagogy to the teaching of programming languages—a research agenda. *SIGCSE Bulletin*, 27(4):9–12, December 1995. doi:10.1145/216511.216517.
- 13 StatisticsTimes.com. Top computer languages, 2020. URL: <http://statisticstimes.com/tech/top-computer-languages.php>.
- 14 L.S. Vygotsky, E. Hanfmann, G. Vakar, and A. Kozulin. *Thought and Language*. The MIT Press. MIT Press, 2012.
- 15 B.C. Wilson and S. Shrock. Contributing to success in an introductory computer science course: A study of twelve factors. In *Proceedings of the Thirty-Second SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '01, page 184–188, New York, NY, USA, 2001. Association for Computing Machinery. doi:10.1145/364447.364581.

Melodic – Teaching Computational Thinking to Visually Impaired Kids

Rui Costa ✉

Centro ALGORITMI, Departamento de Informática,
University of Minho, Campus Gualtar, Braga, Portugal

Cristiana Araújo ✉ 🏠 

Centro ALGORITMI, Departamento de Informática,
University of Minho, Campus Gualtar, Braga, Portugal

Pedro Rangel Henriques ✉ 🏠 

Centro ALGORITMI, Departamento de Informática,
University of Minho, Campus Gualtar, Braga, Portugal

Abstract

This paper presents a proposal, called Melodic, to develop Computational Thinking skills in kids with special educational needs, in this case blindness. The aim of this research is to characterize the subject and identify what are the current most used and best practises to teach this different way of Thinking to kids under those circumstances. In this paper more technical aspects are discussed, such as the architecture for the proposed application in order to accomplish the project goals. Melodic was carefully designed to have an intuitive interface for blind people and a seamless workflow, combining tactile hardware, and QR Code technology with a sound based output.

2012 ACM Subject Classification Social and professional topics → Computational thinking

Keywords and phrases Computational Thinking, Visual Impaired Students Education, Teaching through Music, Learning Resource

Digital Object Identifier 10.4230/OASICS.ICPEC.2021.4

Funding This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020.

Acknowledgements We would like to thank Isabel Barciela and Marta Paço both from Íris Inclusiva for their valuable comments and suggestions; Without their collaboration this work would not be possible.

1 Introduction

In today's world technology is everywhere we look. As a consequence employers look for people with strong skills in computational field. This fact leads many young people to choose programming courses. Programming students normally face a great difficulty due to the traditional way of thinking they are taught in conventional schools, while they should be trained in Computational Thinking that is the key to solve general or programming problems. Computational Thinking is based on some key concepts such as *Logical Reasoning*, *Algorithm Design*, *Decomposition*, *Pattern Recognition*, *Abstraction* and *Evaluation* and changes the whole way of thinking about the resolution of a given problem [10, 3, 12]. This change of the way of thinking is not an easy task so the sooner it is introduced to people the better. That said, it is of uttermost importance to research for a strategy to teach children in a captivating way to keep them motivated. Currently one technique that is very used is “Game Based Learning” [8].

This technique is characterized by being a type of game play with defined learning outcomes. Usually the game used is a digital one, but this is not always the case. There are many arguments in favour of game based learning, such as:



© Rui Costa, Cristiana Araújo, and Pedro Rangel Henriques;
licensed under Creative Commons License CC-BY 4.0

Second International Computer Programming Education Conference (ICPEC 2021).

Editors: Pedro Rangel Henriques, Filipe Portela, Ricardo Queirós, and Alberto Simões; Article No. 4; pp. 4:1–4:14



OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

- **Motivation:** Normally games for entertainment are able to motivate learners to stay engaged over longer periods of time through a series of features of motivational nature. These include incentive artifacts, such as stars, points, leader boards, badges, and trophies, as well as game mechanics and challenges that learners find interesting;
- **Player Engagement:** Is related to motivation and relies on design decisions that reflect the specific learning goal, learner characteristics, and setting;
- **Adaptability:** Learner engagement is facilitated by the adaptive part of the game, being this customizable by the player or personalized. This characteristic is defined by the capability of engage each learner in a way that reflects his or her specific profile;
- **Graceful Failure:** Rather than putting failure as an undesirable outcome, it is an expected and sometimes necessary step in the learning process. This is achieved by lowering the consequences on a failure situation and feedback on the wrong answer so that the player can learn from his or her mistakes.

In order to teach Computational Thinking properly it is mandatory to adapt the Learning Resources to the target learners. Resources must be adequate to the training of Computational Thinking so that the students develop new knowledge and the different abilities and acquisition of values that define Computational Thinking [1, 11].

Reviewing the literature in the area, it is clear that there are some platforms to support the teaching/learning process aimed at a general audience; however, when it concerns special need kids, the available resources are very few. It is very important to realize that most of the referred resources rely upon a visual interface based program paradigm. In order to have a system that successfully helps blind kids to learn the concepts of Computational Thinking, it is mandatory that the system does not rely on a visual interface but on an audio or tactile one [9].

With this in mind, the audio based interface combined with a tactile input device seems the best approach to create resources to teach some Computational Thinking concepts through music. That musical approach has clear advantages, as listed below [6]:

- Debugging a song to make it sound like desired, is similar to debugging a program in order to make it to have the desired behavior;
- Musical phrases correspond to program commands;
- A song is a combination of musical phrases just as a program is a combination of statements.

In this way, we intend to create a Learning Resource capable of, through musical based games, teach the fundamental concepts of Computational Thinking to blind kids to help them to solve programming or general problems using this approach.

The paper is organized as follows. Section 2 explains the concept of Learning Resource, characterizes blind and visually impaired people way of learning, ways to teach visual impaired students and the special characteristics that we have to take in consideration when choosing or creating learning resources for this context. Section 3 presents the principle of the Melodic. Here we can see understand the way of interacting with this application and what kind of output we are expecting to get. Section 4 talks about Melodic's architecture and used technologies. Also in this chapter are described the implementation approaches and prototype testing. Section 5 are given some examples of teaching approaches and ways to use this system to reach its goal, train Computational Thinking. Section 6 summarizes all the work done until this point and the next steps that are predicted to be done for this project.

2 Adaptability of Learning Resources to Blind Students

Learning Resource is a device used for educational purposes in any format, real or virtual, that illustrates or supports one or more elements of a course of study; and may enrich the experience of the pupil or teacher. Learn resources will serve to train Computational Thinking, so they play a very important part in this process. For this reason the resources must be adequate to the training of Computational Thinking so that students can develop new knowledge and the different abilities and acquisition of values that define Computational Thinking [1].

As far back as 1967, there were reported various issues related to the education of visually impaired learners. The focus was on the importance of vision and the mode of reading, and attempted to illustrate how intelligence manifests itself in blind learners as compared to deaf. Their work showed the relevance of blindness and information processing and also illustrated the maximum utilization of available sensory data during learning activities and the translation of visual stimuli [7].

In visual impaired learners conceptual development and abstract thinking seem to be delayed by the absence of visual stimulation or images. With this cognitive development occurs more slowly independently of the age group [5].

A technique to teach conceptual subjects is using tactile stimuli to overcome the difficulty imposed by the lack of sight. Figure 1 illustrates how multiple tactile stimuli can compensate for the loss of sight, allowing learners to perceive size and shape of objects [4]. However, blind people are easily overwhelmed by the complexity of very “full” or “busy” diagrams, sketcher or models. A way to counter this feeling is to be spatially oriented when walking or reading documents.

In the learning point of view, blind and visually impaired learners require specific strategies, addressing their unique learning mediation needs. With loss or absence of vision, the amount of sensory data available to the learner is reduced and for this reason is very important the use of multi-sensory approaches, like the one in Figure 1 [2].

To facilitate this multi-sensory interface, Braille text, talking calculators, computer voice over and many more became standard equipment to teach blind and visually impaired people.

In order to best adapt visually impaired students it is mandatory to adapt the curriculum with strategies such as [4]:

- setting a substitute task of similar scope and demand;
- replacing one impossible or unfriendly task with one of a different kind;
- allowing the learner to undertake the task at a later date;
- using another planned task to assess more outcomes than originally intended;
- giving the learner concessions to complete a task;
- using technology, aides or other special arrangements to undertake assessment tasks;
- using an estimate based on other assessments or work completed by the learner;
- considering the format in which the task is presented.

With this techniques we can adapt conventional teaching to a visually impaired students so that they can learn most of the subjects regarding programming skills.

3 Melodic, the principle and workflow

In this section the fundamental principle of Melodic will be discussed: helping visual impaired people train Computational Thinking. For this we use a set of tactile blocks with different



■ **Figure 1** Combination of three-dimensional models with “Zytec” sketches labelled in Braille as a learning strategy.

shapes, textures and meanings (Figure 2). Each one of this blocks has an assigned value and are part of a class of blocks:

- **Special Block:** This block class is characterized by a smooth surface. Special blocks can be of three different types:
 - **Control:** These blocks control the beginning and the end of an algorithm or instruction. The two blocks shown at the top of Figure 3 are used for beginning and ending sequences. The two blocks shown at the bottom of Figure 3 are used to start and stop loops.
 - **Speed:** These blocks control the time interval between musical notes. As we can see in Figure 4, there are three levels of speed: *slow* (represented by the symbol “>”), *medium* (represented by the symbol “> >”) and *fast* (represented by the symbol “> > >”). The usage of this block is not mandatory; default value is “slow speed”;
 - **Instrument:** These blocks control the instrument that shall be played. The letter G means *Guitar*, P means *Piano*, and F means *Flute* (Figure 5). This block can be used multiple times throughout the sequence creating a set of notes played by different instruments. This is also a non-mandatory block being Piano the default instrument.
- **Number Block:** This block class is characterized by a texture with lines and represent numbers from one to ten. These assign value to the number of iterations of a loop or the number of repetitions of a single note (Figure 6).
- **Musical Note Block:** This block class is characterized by a rough texture and represent musical notes. The output will be a sound representation of these musical notes (Figure 7); The musical notes played are: *Dó*, *Ré*, *Mi*, *Fá*, *Sol*, *Lá*, *Sí*, where *Dó* is represented by the block that contains only *one point*, the *Ré* is represented by the block that contains *two points*, and so on.

As previously mentioned, each block class is characterized by its texture. It is very important that each class of blocks has a different texture so that the visually impaired user can easily distinguish each class. The blind user must first learn the texture that identifies each class of blocks.



■ **Figure 2** Tactile Blocks used to interact with Melodic.



■ **Figure 3** Control Blocks.

4:6 Melodic – Teaching Computational Thinking to Visually Impaired Kids



■ Figure 4 Speed Blocks.



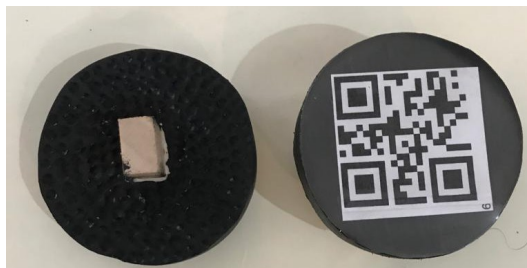
■ Figure 5 Instrument Blocks.



■ Figure 6 Number Blocks.



■ Figure 7 Musical Note Blocks.



■ Figure 8 Top and Bottom faces of one Block.

In addition to the texture, the blocks always have the base painted in black (a dark color) and the symbol of the block in wooden color (light color). This color contrast is very important for low vision users, as it allows them to more easily identify the symbol of each block.

With this tactile blocks the user can create algorithms to represent some melodies, being able to increase in complexity as he improves his/her Computational Thinking skills. Each block contains a QR code at the bottom that identifies its meaning (Figure 8). To read the meaning of each block, the user must install the Melodic application on his smartphone. After that, with using the mobile app the user must read the QR code located at the bottom of the block.

In Section 4 will be presented the architecture of Melodic.

4 Melodic, system architecture

Melodic is a system aimed to be used with a mobile device, being this either a smartphone or tablet. To mitigate incompatibilities between *Apple* and *Android* devices, the *React Native*¹ framework was used, which provides parity between the two platforms with the same JavaScript code.

In Figure 9 we can see Melodic's Architecture Diagram that depicts the system components:

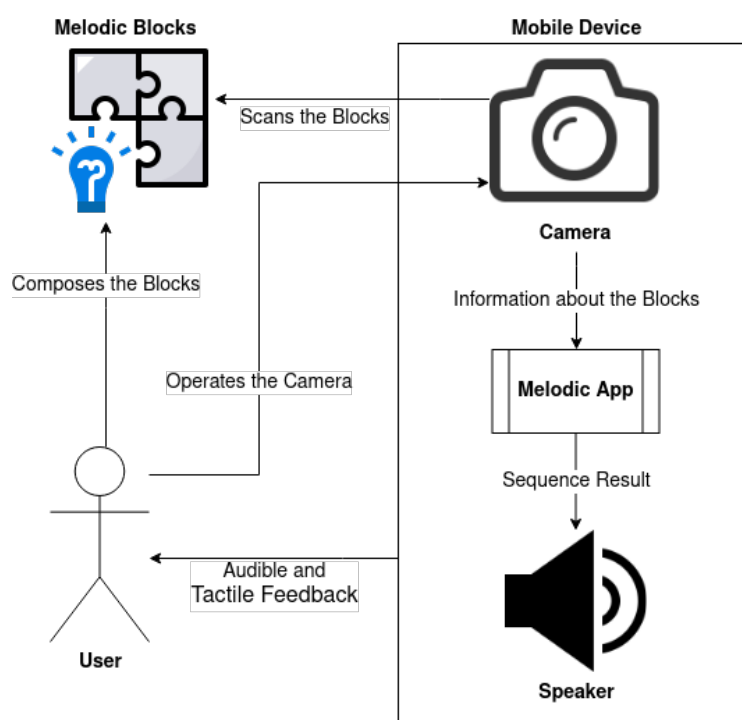
- **User:** The user controls the composition of the blocks to form a musical sequence that produces the desired sound after being scanned by the mobile device;
- **Melodic Blocks:** Set of ordered blocks (algorithm) composed by the user to form the desired musical sequence. Each block has a QR code, at the bottom, so that the mobile device can read its meaning;
- **Mobile Device:** Component in charge of all the logical operations behind this app containing multiple sub-domains:
 - **Camera:** The camera is in charge of reading the QR codes of the blocks;
 - **Melodic App:** It is the core of the system. This module receives, through the camera, the QR code of each block and processes it, behaving like a compiler. After processing the QR codes, the Melodic App sends the information (musical melody) to the Speaker module;
 - **Speaker:** The speaker receives the information, sent by the Melodic App, with the musical melody to play and converts it into a real sound to output to the user.

It is important to notice that the mobile device also provides tactile feedback to the user, in the form of vibrations, to alert the user that the QR code of a given block has been scanned.

In order to facilitate the testing of this system in a easy way, it was developed with the help of Expo². Expo creates a project capable of being accessed by anyone without the need of publishing it on the App Store or Play Store. To access Melodic it is just necessary to install the app Expo.Go and then scan the QR code exhibited Figure 10. Of course this means that the mobile device must be able to read QR codes. To suit this need it is used a library called *expo-barcode-scanner* that provides the necessary functions.

¹ For more information about React Native consult: <https://reactnative.dev/>

² To download Expo consult: <https://expo.io/client>



■ **Figure 9** Melodic's System Architecture Diagram.

After the first tests set succeeded, a new set of tests is being prepared to be experimented with members of **Íris Inclusiva**³.



■ **Figure 10** QR code to access Melodic.

5 Melodic, using the system

After describing Melodic suite, components and interconnection, this section will provide some examples to illustrate how to utilize Melodic as a Learning Resource to train Computational

³ For more information about **Íris Inclusiva** consult: <https://irisinclusiva.pt/>

Thinking. An incremental approach to the complexity of problems presented is advised. For this a collection of different exercises can be designed, each one increasing the complexity level:

- **Simple Algorithm:** This type of algorithm consists of only basic instructions. Figure 11 shows an example of a very simple algorithm composed of four instructions: starting block, two musical note blocks and the ending block.
- **Repetition Algorithm:** This type of algorithm is composed of blocks of basic instructions and blocks of repetition. Figure 12 shows an example of this type of algorithm. In this case, in addition to the starting and ending blocks, we have one musical note block in second position, the block with the number three, and then, another musical note block. The block with the number three is intended to play three times the musical note block preceding the repetition number.
- **Changing the Instrument and Speed Instruction:** In this case, the user can create an algorithm with attributes:
 - **Instrument Played:** With the Instrument Blocks the user can change one or more times the instrument played in the sequence (Figures 13 and 14 respectively). When the instrument block is used, the musical notes after the instrument block are played using the sound of the defined instrument;
 - **Speed between notes:** With the Speed Block the user can change the speed between notes at the time of reproduction. This changes the speed for the whole sequence (Figure 15).
- **Iterative Algorithm (Loop):** In this case the user can program sequences that must be executed one or more times implementing the loop concept. Besides the start and end blocks for the whole algorithm, when the user wants to insert a loop, he must insert also special blocks as follows: start, end, number of iterations and the loop body. In this case, the order of the blocks is very important. Figure 16 shows an example of how to implement a loop sequence. Like all algorithms, the program starts with the starting block. To build the loop, it is necessary to place the Start Loop block, then the number of the iteration, then the body of the loop (composed of several other blocks) and, finally, the End Loop block. The end block determines the end of the algorithm. This type of algorithms is the highest degree of complexity in this system.

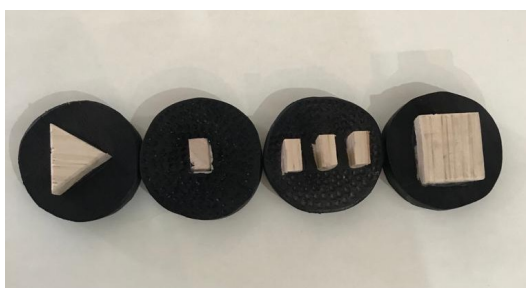
The user must feel free to experiment with other block sequences at his will.

To use the system the user must firstly create the required block sequence. With the sequence prepared the user must open the Melodic App and start scanning the blocks. For each one the user must turn it around, scan it, and turn it back over to avoid QR Code scanning conflicts. At the end of the sequence, when the end sequence block is read the system plays the programmed notes.

To listen the melodies resulting from the presented algorithms please access the project Web site at: <https://epl.di.uminho.pt/~gepl/Melodic>.

In order to obtain a melody, blocks must be organized in a correct sequence. Otherwise Melodic will detect and signalize an error. Regarding wrong block sequences, Melodic copes with 2 different situations, namely:

- **Start Error:** This error occurs when the user forget to initialize the sequence of blocks with the *Start Block*. Figure 17 shows a sequence that misses a *Start Block* causing an error message to be thrown.
- **Loop Error:** In this case, the user composes the blocks to create the loop sequence in a wrong way leading to an incorrect syntax. Figure 18 shows a loop error caused by missing the number of iterations.



■ **Figure 11** Example of a Simple Algorithm.



■ **Figure 12** Example of a Repetition Algorithm.



■ **Figure 13** Example of an Algorithm with a Single Instrument Change.

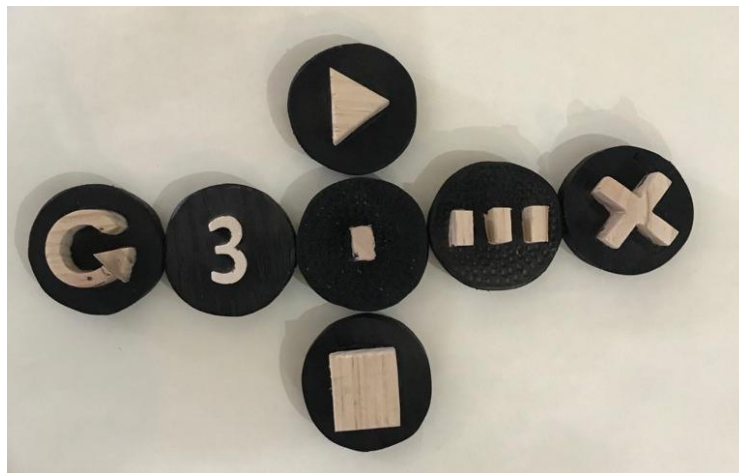


■ **Figure 14** Example of an Algorithm with Multiple Instrument Changes.

To listen to examples of error messages the reader can consult the project homepage at:
<https://epl.di.uminho.pt/~gepl/Melodic>.



■ **Figure 15** Example of an Algorithm with Speed Change.



■ **Figure 16** Example of an Iterative Algorithm (Loop).

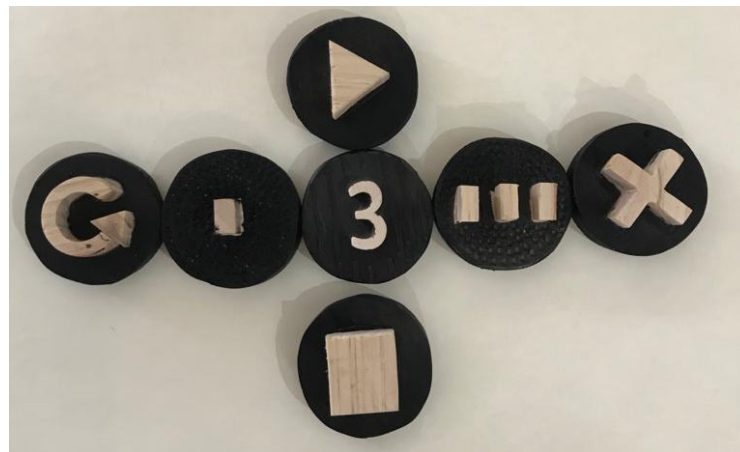


■ **Figure 17** Example of a Start Block Error.

6 Conclusion

This paper describes Melodic, the motivations behind it and all its features. Melodic is a mixed⁴ serious game intended to be used as an Learning Resource to train Computational Thinking adapted to visual impaired people.

⁴ In the sense that it is partially unplugged but also it uses a software application to complement its functionality.



■ **Figure 18** Example of a Loop Creation Error.

The paper starts by an introduction to the Computational Thinking subject and how it is important to the 21.st Century youth. Following this is an explanation of some methods on how to adapt learning resources to the visual impaired community, focusing on young learners and multiple sensory platforms. Next is the introduction to Melodic, it's principle and workflow, explaining how, with the help of tactile blocks specifically designed for this system, we can train Computational Thinking concepts to visual impaired students. Then Melodic's Architecture is presented. In this section the design and all technological details of the system implementation are described. Finally a description of how to use the system illustrated by simple examples is made.

A core component of Melodic is the set of the tactile blocks. This was one of the biggest difficulties of this project due to our lack of knowledge in the area of visually impaired people. It was here that Íris Inclusiva came to help. They gave us a lot of valuable opinions and advises on what should the block's shapes and textures be. This guided us into the final system design described previously in this document.

Melodic is also thought to be available to everybody, being an institution or an individual person, who wants to have a set of blocks for their own. For this, the person can use his creativity and build each block using materials such as: wood, styrofoam, clay, among others. After having all the blocks ready, it is necessary to paste the respective QR codes at the bottom of each block. These QR codes are all labeled in one document accessible online at <https://epl.di.uminho.pt/~gepl/Melodic>.

Melodic is an inclusive Learning Resource because although it is thought to be used by blind or low vision students, it can also be used by students who do not have visual difficulties. This is due to the fact that blocks use a general and intuitive alphabet and do not resort to a blind people specialized one.

In conclusion Melodic aims to help visually impaired people to get in touch with a reality that normally relies on a visual interface, that is programming, enabling them to train Computational Thinking.

In order to prove that Melodic works, a preliminary test was performed. For this Íris Inclusiva was a major help providing the space and the user to be the first visual impaired to test the system.

In this test we could prove that the blocks are easily identifiable (Figure 19) and that the user could in few minutes get very proficient in reading the block's QR codes (Figure 20).

To proceed with Melodic testing, a list of people to test is being developed in partnership with Íris Inclusiva. This tests is a proof of concept of the system proposed.

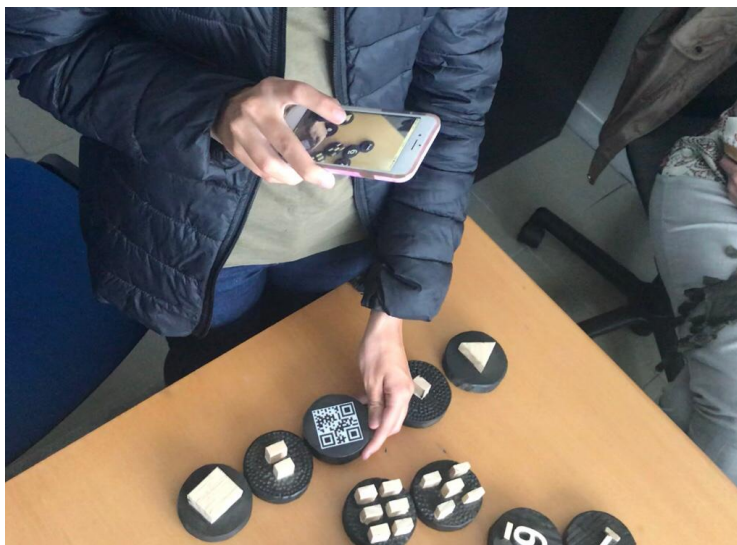
To access the effectiveness of Melodic, two main issue have to be proven:

- Visual Impaired people can interact with the system;
- The system develops Computational Thinking.

The first one has already been tested and for the second one, diagnostic and final tests are being designed. These tests have the objective to evaluate the student's capability to solve problems before and after using Melodic, to understand if their Computational Thinking skill were improved.



■ **Figure 19** User Finding the Tactile Blocks.



■ **Figure 20** User Reading the Sequence Tactile Blocks QR Codes.

As future work, it is necessary to test the application with more blind and low vision students. For that, we will design and conduct some experiments with impaired members of Íris Inclusiva association. Then, we will make possible changes identified in the testing phase. Finally other more sophisticated experiments will be conducted with other groups.

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 Dr. Elizabeth J. Erwin, Tiffany S. Perkins, Jennifer Ayala, Michelle Fine, and Ellen Rubin. “you don’t have to be sighted to be a scientist, do you?” issues and outcomes in science education. *Journal of Visual Impairment & Blindness*, 95(6):338–352, 2001. doi:10.1177/0145482X0109500603.
- 3 Council for The Curriculum Examinations and Assessment. Computing at school: Northern ireland curriculum guide for post primary schools. Technical report, CCEA, 2018.
- 4 William Fraser and Mbulaheni Maguvhe. Teaching life sciences to blind and visually impaired learners. *Journal of Biological Education - J BIOL EDUC*, 42:84–89, March 2008. doi:10.1080/00219266.2008.9656116.
- 5 J. Freeman. *The Psychology of Gifted Children*. Wiley Series in Developmental Psychology and Its Applications. Wiley, 1985. URL: <https://books.google.pt/books?id=dtB9AAAAAAAJ>.
- 6 Mark Guzdial. Teaching programming with music: An approach to teaching young students about logo. *Logo Foundation*, 1991.
- 7 N.G. Haring and R.L. Schiefelbusch. *Methods in Special Education*. McGraw-Hill series in education: Psychology and human development in education. McGraw-Hill, 1967. URL: <https://books.google.pt/books?id=ADQ1AAAAAAAJ>.
- 8 Jan L. Plass, Bruce D. Homer, and Charles K. Kinzer. Foundations of game-based learning. *Educational Psychologist*, 50(4):258–283, 2015. doi:10.1080/00461520.2015.1122533.
- 9 Alpay Sabuncuoğlu. Tangible music programming blocks for visually impaired children. In *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '20, page 423–429, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3374920.3374939.
- 10 Victor Silva, Heleniara Moura, Suelen Paula, and Ângelo Jesus. Algo+ritmo: Uma proposta desplugada com a música para auxiliar no desenvolvimento do pensamento computacional. In *Anais do XXV Workshop de Informática na Escola*, pages 404–413, Porto Alegre, RS, Brasil, 2019. SBC. doi:10.5753/cbie.wie.2019.404.
- 11 Salete Teixeira, Diana Barbosa, Cristiana Araújo, and Pedro Rangel Henriques. Improving Game-Based Learning Experience Through Game Appropriation. In Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões, editors, *First International Computer Programming Education Conference (ICPEC 2020)*, volume 81 of *OpenAccess Series in Informatics (OASICs)*, pages 27:1–27:10, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/OASICs.ICPEC.2020.27.
- 12 Stephen Wolfram. How to teach computational thinking, 2016. URL: <https://writings.stephenwolfram.com/2016/09/how-to-teach-computational-thinking/>.

An Open-Source Gamified Programming Learning Environment

José Carlos Paiva  

CRACS - INESC-Porto LA, Portugal

DCC - FCUP, Porto, Portugal

Ricardo Queirós   

CRACS - INESC-Porto LA, Portugal

uniMAD - ESMAD, Polytechnic of Porto, Portugal

José Paulo Leal   

CRACS - INESC-Porto LA, Portugal

DCC - FCUP, Porto, Portugal

Jakub Swacha   

University of Szczecin, Poland

Filip Miernik  

University of Szczecin, Poland

Abstract

The importance of e-learning tools facilitating the process of learning to program is growing, especially as the pandemic-caused lockdown enforced distance learning in many countries. The key success factor in this process is the provision of an instant and relevant feedback to students. In this paper, we describe a novel open-source programming learning environment featuring automatic assessment of students' solutions and customized gamification. This environment has been developed as a part of the FGPE framework.

2012 ACM Subject Classification Applied computing → Computer-managed instruction; Applied computing → Interactive learning environments; Applied computing → E-learning

Keywords and phrases learning environment, programming exercises, gamification, programming learning, automatic assessment

Digital Object Identifier 10.4230/OASICS.ICPEEC.2021.5

Category Short Paper

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).

1 Introduction

The role of feedback in improving knowledge and skill acquisition is considered crucial ([29] and works cited therein). In programming education, particularly, an instant feedback can be generated automatically based on the submission code, providing the student with relevant information about general concepts, task constraints, mistakes made, hints on how to proceed or even meta-cognition [16].

There is also a major concern amongst educational researchers with the disengagement of the students from the learning activities, which frequently leads to academic failure and, lastly, dropout. This issue is even more noticeable in distance learning [28], which became the default way of learning in many countries in times of the current pandemic. One way to counteract this problem is through gamification, which has been proven as a capable tool to reduce the dropout rate [10].



© José Carlos Paiva, Ricardo Queirós, José Paulo Leal, Jakub Swacha, and Filip Miernik; licensed under Creative Commons License CC-BY 4.0

Second International Computer Programming Education Conference (ICPEEC 2021).

Editors: Pedro Rangel Henriques, Filipe Portela, Ricardo Queirós, and Alberto Simões; Article No. 5; pp. 5:1–5:8

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

5:2 An Open-Source Gamified Programming Learning Environment

In this paper, we describe a novel open-source programming learning environment addressing both the above-mentioned needs, featuring automatic assessment of students' solutions and customized gamification. As the environment has been developed within the Framework for Gamified Programming Education (FGPE) project [9], we called it FGPE PLE.

The rest of this paper is organized as follows. Section 2 reviews the current state of gamification in education, focusing on empirical studies describing experiments which applied game elements into a learning environment. Section 3 presents the FGPE PLE, its architecture and user interface. Finally, Section 4 summarizes the contributions of this work and indicates some future directions.

2 Related Work

Gamification of education has been a top trending in the last years [30], which can be explained by the positive boost in motivation and user experience it is accredited with. There are several case studies in the literature, particularly performed in high school and universities, covering different learning subjects, ranging from Computer Science / Information Technology (CS/IT) [34, 1, 2, 4, 11, 22, 33] to Mathematics (Maths) [8, 35, 6, 25], Foreign Languages (FL) [13, 26], Communication and Multimedia (C&M) [3, 12, 14, 15], and Medicine / Biology (M/B) [27, 20, 5]. A search for experimental studies that apply gamification methods in educational learning environments was conducted in Google Scholar and ScienceDirect databases, using combinations of the following keywords: gamification, serious games, education, and learning. From the returned results, only studies published since 2014 which have reported findings in high school and university subjects of the previously mentioned areas were considered. The result was a set of about 50 empirical research studies, of which 20 were selected according to their relevance. Table 1 presents a comparative study of the collected research papers, regarding the applied game elements.

■ **Table 1** Game elements reported in the selected research papers.

Game Elements	CS/IT							Maths				FL		C&M			M/B				
	[34]	[1]	[2]	[4]	[11]	[22]	[33]	[8]	[35]	[6]	[25]	[13]	[26]	[3]	[12]	[14]	[15]	[27]	[20]	[5]	
Badges	✓	✓	✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓				✓	
Collaboration	✓			✓			✓	✓				✓		✓	✓	✓				✓	✓
Direct Competition ¹							✓														
Content Disclosure		✓		✓		✓	✓							✓							
Exhaustible Resources								✓		✓	✓			✓	✓		✓				
Leaderboards	✓	✓		✓		✓		✓	✓	✓	✓		✓	✓	✓	✓				✓	✓
Levels							✓					✓		✓	✓	✓	✓			✓	
Points ²	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓
Progress Indicator	✓	✓		✓								✓		✓	✓	✓					✓
Quests							✓					✓		✓	✓						✓
Real-world Rewards																					✓
Serious Games / Immersion							✓	✓	✓		✓		✓	✓	✓						✓
SLP ³			✓			✓					✓			✓	✓						
Status	✓					✓		✓													✓

¹Only core competition, i.e., one player (or team) plays against other, if one wins, the other loses.

²Includes experience points (XP), skill points (score), influence points (rating, reputation), and automatic grades.

³Similar Learning Path (SLP) consists of using data from other students to predict and display the current progress of a student (not leaderboards).

From this study, we can observe that badges, points, and leaderboards are the most applied game elements in general. This was expected as those are the easiest elements to apply, and reuse in distinct educational scenarios. Only a few works use, for instance, quests (5), content disclosure (5), status (4), and direct competition (1) as these are harder to implement and less reusable. Surprisingly, even though there is much more work on gamification of CS/IT education compared to other areas of education, no significant difference in game elements could be found.

Regarding experimental results, the effects on students are mostly slightly positive or neutral, with a few exceptions [12, 6]. Nevertheless, in the area of CS/IT the results are overall positive, i.e., they either promote engagement, participation, dedicated time, or learning outcome.

3 Programming Learning Environment

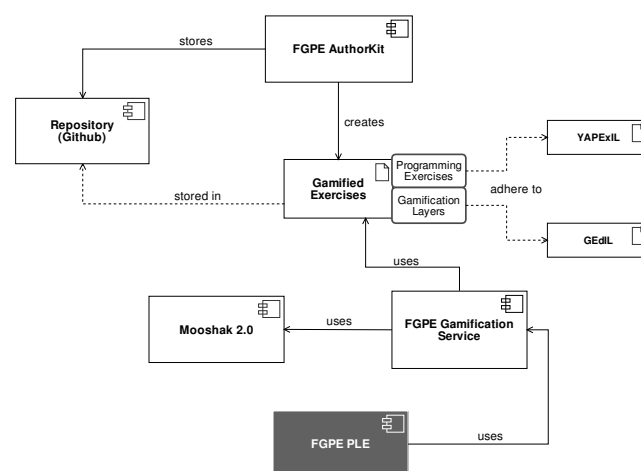
FGPE Programming Learning Environment (PLE) is an open-source progressive web application designed to engage learners while learning to program. To this end, it combines automated assessment with a meaningful composition of gamification elements, both provided by the Framework for Gamified Programming Education (FGPE) ecosystem. The PLE is effortlessly reusable across distinct courses and programming languages, being independent of the gamification layer linking the activities.

The following subsections provide an in-depth overview of the FGPE PLE. Subsection 3.1 details the architecture of the environment. Subsection 3.2 presents its user interface.

3.1 Architecture

The FGPE PLE is a web-based environment that aims to deliver a single interface to a complete ecosystem for gamified programming education. This PLE handles two types of users: students – who register and play the games (i.e., courses) in which they are enrolled, by solving the proposed challenges – and teachers – who manage the courses, enroll students, assign them into groups, and collect/analyze data about their progress. As users must also be able to access other tools of the same ecosystem, particularly authors, and use different authentication mechanisms, authentication and authorization is managed by Keycloak [17]. Keycloak is an open-source identity and access management solution providing numerous features for the applications, such as centralized user management, authentication, single sign-on and identity brokering, and user federation and social login.

Figure 1 presents the FGPE ecosystem, in which the FGPE PLE plays a key role. This ecosystem is composed of two formats, one for describing programming exercises – YAPExIL [24] – and another for gamification layers – GEdIL [31], three tools including an authoring tool, a gamification service, and an evaluation engine, and GitHub.



■ **Figure 1** Architecture of the FGPE ecosystem, highlighting the FGPE PLE.

5:4 An Open-Source Gamified Programming Learning Environment

The FGPE ecosystem supports seven distinct types of programming exercises, in particular: `BLANK_SHEET`, which asks the students to develop their solution from scratch; `EXTENSION`, which presents a partial solution for the student to complete; `IMPROVEMENT`, which provides a correct initial source code that needs some optimization; `BUG_FIX`, that gives a buggy solution for the student to find the right code; `FILL_IN_GAPS`, which provides code with missing parts; `SPOT_BUG`, that asks students to indicate the location of the bugs in a buggy solution; and `SORT_BLOCKS`, which asks students to sort the several blocks of code of a solution. To this end, programming exercises are defined in YAPExIL [24], a novel JSON format introduced to address the lack of support for non-traditional programming exercises in existing formats. Furthermore, YAPExIL does not target a specific evaluation engine, heading all efforts to maximize expressiveness and facilitate the conversion from/to other programming exercise formats.

Gamification layers are specified separately from programming exercises using GEdIL [31], an open format for the specification of gamification layers for educational contents. Even though it does not target a specific course, GEdIL was initially designed to cover particular requirements of gamification for programming courses [32], including a vast collection of rewarding mechanisms, such as points, badges, virtual items, and leaderboards to provide extrinsic motivation, as well as unlockable and secret content, different activity modes (e.g., speedup and duels), among others. The game dynamics and mechanics are, then, parsed and applied in the referenced collection of activities by the gamification service.

FGPE Gamification Service [21] is a GraphQL service designed to fulfil this goal, consuming a GEdIL layer and transforming it into a game, played by enrolled students. Hence, the service has complete support for various rewarding mechanisms such as points, badges, coupons, virtual items, leaderboards, locked and secret content, and different activity modes (e.g., time-bomb and speedup). Furthermore, the FGPE Gamification Service is the single point of access of the FGPE PLE to this ecosystem. Consequently, it should not only apply gamification rules, but also deliver the challenge and activity statements as well as handle the automated assessment of activities. To this end, the service uses plugins, i.e., consumers of the evaluation engine APIs, that leverage on the evaluation engine connected that is adequate to assess that type of activity.

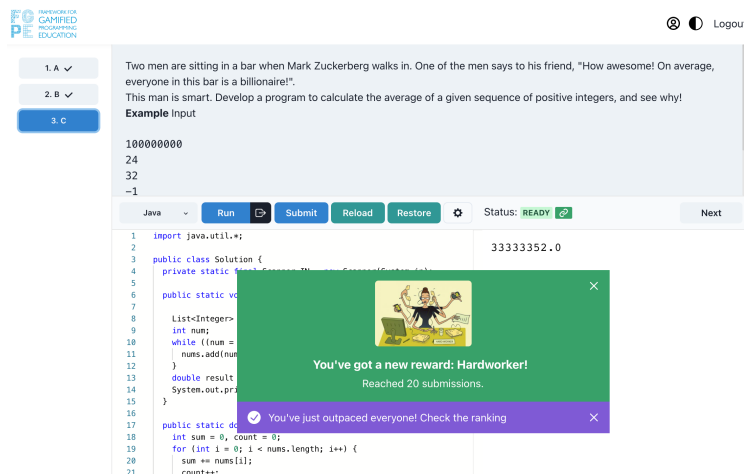
An automated evaluation engine guarantees accurate and timely feedback to students. Such an engine is responsible for marking and grading exercises. In this ecosystem, an evaluation engine will: (1) receive the identification of the student, a reference to the exercise, and the student's attempt to solve it; (2) load the exercise by reference from the repository (if necessary); (3) compile the solution and run the tests against the student's program, comparing the obtained output to the expected; and (4) build report of the evaluation with passed/failed test cases, the grade, and, possibly, some feedback. Therefore, Mooshak 2 has been selected. Mooshak [18] is an open-source web-based system for managing programming contests, which includes automatic judging of submitted programs. Version 2 inherits all features from its original version, improving feedback and attaching support for different types of exercises. Moreover, Mooshak supports custom static and dynamic analysis scripts which enables assessment for the non-traditional programming exercises previously described.

Finally, having defined two new formats, this ecosystem required a tool to facilitate the authoring of new programming exercises adhering to and importing exercises in existing formats into such new programming exercise format as well as their connection with gamification rules. FGPE AuthorKit [23] is a web application designed to support the authors through the entire process of preparing gamified programming exercises, including (1) creation of exercises with their associated metadata, (2) design of the gamification techniques for a

specific exercise or their collection, (3) definition of the content structure and sequencing rules, and (4) importing and exporting the content in several popular programming exercise formats, using YAPEXIL and GEdIL as the base. The user can share content with other peers, internally or via a GitHub repository where all exercise data is synchronized.

3.2 User Interface

The FGPE PLE is a progressive web application developed in ReactJS [7] – “a JavaScript library for building user interfaces” –, meaning that it is an application software delivered through the web intended to work on any platform that utilizes a standards-compliant browser, including both desktop and mobile devices. It uses the Apollo Client, a community-driven effort to build an easy-to-understand, flexible, and powerful GraphQL client, to manage and consume remote data from the FGPE Gamification Service without worrying about infrastructure code for networking and caching.



■ **Figure 2** Exemplary screenshot of the FGPE PLE User interface.

The User Interface (UI), whose exemplary view is presented in Figure 2, follows a component-driven development approach, which consists in dissecting and splitting the interface into small repeatable patterns, develop these patterns, and join them together to form larger components and pages. The main components of the PLE are the following.

Code Editor is based on Monaco Editor [19], the editor that powers Visual Studio Code.

This editor allows students to code in almost any programming language, starting from a skeleton provided by the exercise author, taking advantage of a vast set of features such as syntax highlighting, parameter hints, smart code navigation, and code completion. For challenges that can be solved in more than one programming language, a language switch is attached at the top of the editor allowing the student to choose the programming language of the editor.

Console is where the results and feedback from executions and submissions is presented. The execution consists in taking the inputs provided by the student through a popup and running the program against these inputs. Submissions follow a similar process but run against the complete set of test cases provided by the exercise author. Unlike an execution, a submission is considered for statistics and can unlock new resources of the course.

Statement Viewer is where the activity statement is displayed. This component can display either Markdown, HTML, or raw text files.

Leaderboard is the component responsible for displaying the usernames and scores, sorted according to certain metrics. Leaderboards can be challenge-scoped or course-scoped, use any of the available metrics to sort users by, and optionally have group visibility.

Push Notifications are small rectangle boxes displayed based on events received from registered GraphQL subscriptions. For instance, received rewards, results of both processed submissions and validations, and other updates.

Profile is the student's space to show off her achievements, including badges, virtual items, experience points, leaderboards' top positions, and course progress.

4 Conclusion

Although gamification has its effectiveness demonstrated in engaging students with learning activities, to the best of the authors' knowledge, until now there has been no programming learning environment that would be, at the same time, open-source, reusable across distinct courses on various programming languages, and enabling instructors to compose their own gamification layer providing a good selection of components to choose from.

This paper introduces a novel programming learning environment having all the traits mentioned above. It is implemented as a progressive web application that constitutes a single interface to a complete ecosystem for gamified programming education developed within the Framework for Gamified Programming Education project [9]; particularly, it renders exercises in the YAPEXIL format [24] and connects with the FGPE GS [21], which processes students' submissions and gamification rules defined in the GEdIL [31] format, generating a relevant instant feedback to the students. Both the programming exercises and gamification layers can be created with the companion FGPE AuthorKit tool [23].

The work described here constitutes the final artifact to complete the ecosystem proposed by the Framework for Gamified Programming Education project [9]. A complete validation of the work developed in this project with a large group of students is the immediate next step in our agenda. Furthermore, the FGPE project will have its continuation in the FGPE Plus project, which aims to make the PLE embeddable in any LTI-compliant Learning Management System as well as enhance the PLE's user experience on mobile devices.

References

- 1 Paul E. Anderson, Thomas Nash, and Renée McCauley. Facilitating Programming Success in Data Science Courses Through Gamified Scaffolding and Learn2Mine. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '15*, pages 99–104, New York, NY, USA, 2015. ACM. doi:10.1145/2729094.2742597.
- 2 Tapio Auvinen, Lasse Hakulinen, and Lauri Malmi. Increasing students' awareness of their behavior in online learning environments with visualizations and achievement badges. *IEEE Transactions on Learning Technologies*, 8(3):261–273, 2015. doi:10.1109/tlt.2015.2441718.
- 3 Gabriel Barata, Sandra Gama, Joaquim Jorge, and Daniel Gonçalves. Studying student differentiation in gamified education: A long-term study. *Computers in Human Behavior*, 71(Supplement C):550–585, 2017. doi:10.1016/j.chb.2016.08.049.
- 4 Andrija Bernik, Goran Bubas, and Danijel Radosevic. A Pilot Study of the Influence of Gamification on the Effectiveness of an e-Learning Course. In *Central European Conference on Information and Intelligent Systems*, pages 73–79, Varaždin, Croatia, 2015. University of Zagreb, Faculty of Organization and Informatics.

- 5 Mads T. Bonde, Guido Makransky, Jakob Wandall, Mette V. Larsen, Mikkel Morsing, Hanne Jarmer, and Morten O. A. Sommer. Improving biotech education through gamified laboratory simulations. *Nature biotechnology*, 32(7):694–697, 2014. doi:10.1038/nbt.2955.
- 6 Katheryn R. Christy and Jesse Fox. Leaderboards in a virtual classroom: A test of stereotype threat and social comparison explanations for women’s math performance. *Computers & Education*, 78(Supplement C):66–77, 2014. doi:10.1016/j.compedu.2014.05.005.
- 7 Facebook. React: A JavaScript library for building user interfaces. <https://reactjs.org>, 2021. accessed on 16 Jan 2021.
- 8 Usef Faghihi, Albert Brautigam, Kris Jorgenson, David Martin, Angela Brown, Elizabeth Measures, and Sioui Maldonado-Bouchard. How gamification applies for educational purpose specially with college algebra. *Procedia Computer Science*, 41(Supplement C):182–187, 2014. 5th Annual International Conference on Biologically Inspired Cognitive Architectures, 2014 BICA. doi:10.1016/j.procs.2014.11.102.
- 9 FGPE Consortium. Framework for Gamified Programming Education. <https://fgpe.usz.edu.pl>, 2018. accessed on 19 Mar 2021.
- 10 Wad Ghaban and Robert Hendley. How Different Personalities Benefit From Gamification. *Interacting with Computers*, 31(2):138–153, March 2019. doi:10.1093/iwc/iwz009.
- 11 Lasse Hakulinen, Tapio Auvinen, and Ari Korhonen. The Effect of Achievement Badges on Students’ Behavior: An Empirical Study in a University-Level Computer Science Course. *International Journal of Emerging Technologies in Learning*, 10(1):18–29, 2015. doi:10.3991/ijet.v10i1.4221.
- 12 Michael D. Hanus and Jesse Fox. Assessing the effects of gamification in the classroom: A longitudinal study on intrinsic motivation, social comparison, satisfaction, effort, and academic performance. *Computers & Education*, 80:152–161, 2015. doi:10.1016/j.compedu.2014.08.019.
- 13 Tatsuhito Hasegawa, Makoto Koshino, and Hiromi Ban. An English vocabulary learning support system for the learner’s sustainable motivation. *SpringerPlus*, 4(1):99, 2015. doi:10.1186/s40064-015-0792-2.
- 14 Caitlin Holman, Stephen J. Aguilar, Adam Levick, Jeff Stern, Benjamin Plummer, and Barry Fishman. Planning for Success: How Students Use a Grade Prediction Tool to Win Their Classes. In *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge*, pages 260–264, New York, NY, USA, 2015. ACM. doi:10.1145/2723576.2723632.
- 15 Jincheul Jang, Jason J. Y. Park, and Mun Y. Yi. Gamification of online learning. In Cristina Conati, Neil Heffernan, Antonija Mitrovic, and M. Felisa Verdejo, editors, *Artificial Intelligence in Education*, pages 646–649, Cham, 2015. Springer International Publishing.
- 16 Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. Towards a Systematic Review of Automated Feedback Generation for Programming Exercises. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, pages 41–46, Arequipa Peru, 2016. ACM. doi:10.1145/2899415.2899422.
- 17 Keycloak. Keycloak: open source identity and access management solution. <https://www.keycloak.org>, 2014. accessed on 9 Jan 2021.
- 18 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.
- 19 Microsoft. Monaco Editor. <https://microsoft.github.io/monaco-editor/>, 2021. accessed on 16 Jan 2021.
- 20 Christa R Nevin, Andrew O Westfall, J Martin Rodriguez, Donald M Dempsey, Andrea Cherrington, Brita Roy, Mukesh Patel, and James H Willig. Gamification as a tool for enhancing graduate medical education. *Postgraduate Medical Journal*, 90(1070):685–693, 2014. doi:10.1136/postgradmedj-2013-132486.
- 21 José Carlos Paiva, Alicja Haraszczuk, Ricardo Queirós, José Paulo Leal, Jakub Swacha, and Sokol Kosta. FGPE Gamification Service: A GraphQL Service to Gamify Online Education.

- In *Proceedings of the 9th World Conference on Information Systems and Technologies*, Terceira Island, Azores, Portugal, 2021. Springer. (to appear).
- 22 José Carlos Paiva, José Paulo Leal, and Ricardo Alexandre Queirós. *Enki: A Pedagogical Services Aggregator for Learning Programming Languages*, page 332–337. ACM, New York, NY, USA, 2016.
 - 23 José Carlos Paiva, Ricardo Queirós, José Paulo Leal, and Jakub Swacha. FGPE AuthorKit – A Tool for Authoring Gamified Programming Educational Content. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '20, page 564, New York, NY, USA, 2020. ACM. doi:10.1145/3341525.3393978.
 - 24 José Carlos Paiva, Ricardo Queirós, José Paulo Leal, and Jakub Swacha. Yet Another Programming Exercises Interoperability Language (Short Paper). In Alberto Simões, Pedro Rangel Henriques, and Ricardo Queirós, editors, *9th Symposium on Languages, Applications and Technologies (SLATE 2020)*, volume 83 of *OpenAccess Series in Informatics (OASISs)*, pages 14:1–14:8, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/OASISs.SLATE.2020.14.
 - 25 Mads Kock Pedersen, Anette Svenningsen, Niels Bonderup Dohn, Andreas Lieberoth, and Jacob Sherson. DiffGame: Game-based Mathematics Learning for Physics. *Procedia - Social and Behavioral Sciences*, 228(Supplement C):316–322, 2016. 2nd International Conference on Higher Education Advances, HEAd'16, 21-23 June 2016, València, Spain. doi:10.1016/j.sbspro.2016.07.047.
 - 26 Bernadette Perry. Gamifying French Language Learning: A Case Study Examining a Quest-based, Augmented Reality Mobile Learning-tool. *Procedia - Social and Behavioral Sciences*, 174(Supplement C):2308–2315, 2015. International Conference on New Horizons in Education, INTE 2014, 25-27 June 2014, Paris, France. doi:10.1016/j.sbspro.2015.01.892.
 - 27 Robin K. Pettit, Lise McCoy, Marjorie Kinney, and Frederic N. Schwartz. Student perceptions of gamified audience response system interactions in large group lectures and via lecture capture technology. *BMC medical education*, 15(1):92, 2015. doi:10.1186/s12909-015-0373-7.
 - 28 Bardh Prenkaj, Giovanni Stilo, and Lorenzo Madeddu. Challenges and Solutions to the Student Dropout Prediction Problem in Online Courses. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 3513–3514, New York, NY, USA, 2020. ACM. doi:10.1145/3340531.3412172.
 - 29 Valerie J. Shute. Focus on Formative Feedback. *Review of Educational Research*, 78(1):153–189, 2008. doi:10.3102/0034654307313795.
 - 30 Jakub Swacha. State of Research on Gamification in Education: A Bibliometric Survey. *Education Sciences*, 11(2):69, 2021. doi:10.3390/educsci11020069.
 - 31 Jakub Swacha, José Carlos Paiva, José Paulo Leal, Ricardo Queirós, Raffaele Montella, and Sokol Kosta. GEdIL–Gamified Education Interoperability Language. *Information*, 11(6):287, May 2020. doi:10.3390/info11060287.
 - 32 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.
 - 33 Alexandru Topîrceanu. Gamified learning: A role-playing approach to increase student in-class motivation. *Procedia Computer Science*, 112(Supplement C):41–50, 2017. Proceedings of KES 2017, Marseille, France. doi:10.1016/j.procs.2017.08.017.
 - 34 Andika Y. Utomo, Afifa Amriani, Alham F. Aji, Fatin R. N. Wahidah, and Kasiyah M. Junus. Gamified E-learning model based on community of inquiry. In *Advanced Computer Science and Information Systems (ICACSIS), 2014 International Conference on*, pages 474–480, Jakarta, Indonesia, 2014. IEEE, IEEE. doi:10.1109/icacsis.2014.7065830.
 - 35 Ibrahim Yildirim. The effects of gamification-based teaching practices on student achievement and students' attitudes toward lessons. *The Internet and Higher Education*, 33(Supplement C):86–92, 2017. doi:10.1016/j.iheduc.2017.02.002.

Integrating a Graph Builder into Python Tutor

Diogo Soares ✉

University of Minho, Braga Portugal

Maria João Varanda Pereira¹ ✉ 🏠 

Research Centre in Digitalization and Intelligent Robotics,
Polytechnic Institute of Bragança, Portugal

Pedro Rangel Henriques ✉ 🏠 

University of Minho, Braga, Portugal

Abstract

Analysing unknown source code to comprehend it is quite hard and expensive task. Therefore, the Program Comprehension (PC) subject has always been an area of interest as it helps to realize how a program works by identifying the code that implements each functionality. This means being able to map the problem domain with the program domain. PC is a complex area, but its importance for programmers is so high that many approaches and tools were proposed along the last two decades. Program Animation is one of those approaches requiring specialized techniques.

For each programming language, there are already tools that enable us to execute a program step by step, visualize its execution path, observe the effect of each instruction on its data structures, and inspect the value of its variables at any point.

In the present context, we sustain the idea that PC techniques and tools can also be of great value for students taking the first steps in programming using a specific language. To this end, we aim to improve Python Tutor, a well-known program visualization tool, with graph-based representations of source code such as Control Flow Graph (CFG), Data Flow Graph (DFG), Function Call Graph (FCG) and System Control Graph (SCG).

This helps novice programmers to understand the source code analyzing not only the variable contents but also a set of automatically generated graph-based visualizations, that were not included in Python Tutor so far. This will allow the students to be focused on certain aspects of the program (depending on the graph), abstracting others such as details of its syntax.

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

Keywords and phrases Program Visualization, Python Tutor, Data Flow Graphs, Control Flow Graphs

Digital Object Identifier 10.4230/OASICS.ICPEC.2021.6

Supplementary Material *Software (Graph Builder):*

<https://graph-builder.di.uminho.pt/visualize.html>

Funding This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the Projects Scopes: UIDB/05757/2020 and UIDB/00319/2020.

1 Introduction

There's along the web many tutorials on programming and several learning methodologies but the majority of them uses the same formula [4]: start with the explanation of a set of examples and propose some similar exercises that train the students to write similar code to solve them. To fully understand a programming language, it is not enough to read the code

¹ to mark corresponding author



of programs written in that language, it is also necessary to comprehend how the computer behaves when executing the programs code. For each piece of code, the student shall be able to build a full map between the program domain and the problem domain.

Usually it is hard to understand his own code, even to a software engineer that has to maintain third-part code he never saw before, the task is much harder. The average developer spends about 60% to 90% of his time understanding code previously developed; these figures are not changing over time [19].

With this in mind, this paper describes and discusses the integration into Python Tutor of a new tool for automatic generation of graph-based visualizations to turn ease the task of program comprehension. As said in [7], the best way to understand programs is by *giving programs another aspect than that of their source code*. This new feature works as Python Tutor plugin and it analyses the input code and produces several types of complementary graphs depending on the user desires. As the program comprehension task is also very relevant for the beginners wishing to understand how a given program works and how it solves a given problem, we believe that Python Tutor improved with our plug-in will be very useful to help programming students.

The paper has 5 sections after that. The state of the art on program animation appears in Section 2. Section 3 discusses how graph-based visualizations can be included in an Animator adding actual value to it. Section 4 exhibits the architecture to build the new tool and discusses how it can be developed and integrated into the host tool. Section 5 illustrates the final system built and describes a experiment conducted to assess the value added by the visualizations provided. Section 6 closes the paper and points out some directions for future research.

2 Program Visualization and Animation

Think-aloud protocol [22] is a method where a person verbalize his thoughts and it can be used not only to improve self–understanding and reasoning but also to explain the cognitive process to others. That’s why is so important to produce documentation when developing software. The use of this method also allows realizing that the thoughts of a developer differ according to the familiarity of a program’s domains [17]. With this in mind, it was derived two concepts of comprehension models, the top-down comprehension model and the bottom-up comprehension models. The top-down approach is used when the programmer is familiar with the program he’s trying to understand. He starts by creating a general hypotheses about the program goal and how it is achieved by using his previous knowledge about the program. This is called the *expectation-based* comprehension, where the programmer has pre-generated expectations of the code’s meaning [13]. After creating his hypotheses, the developer starts searching in the code for algorithms/structures that he can link to his theory and refines his hypotheses as he does that [6]. This is the *inference-based comprehension* [13]. If the programmer has no previous knowledge about a program, he has to analyze line by line to create a hypothesis about the program purpose. In this method, the developer starts aggregating functions by their goals. What usually happens is that developers end by using an integrated model [11] where they use top-down approach when possible and bottom-up when necessary.

Although the way of coding varies from person to person, follow a guide of good practices in programming (i.e. appropriate variable names and indentation rules [18]) can lead to an uniformed structure that makes program comprehension easier. A different form of coding can really slow down the comprehension of a program for the most experienced programmer [20].

Nowadays, there's even more factors that can affect program comprehension. For example, a simple color schema, available in syntax-oriented text editor/IDE, helps the programmers comprehending a program [16].

Newer studies have their focus not only on source code comprehension, but also on the comprehension of structures, hierarchies and architecture of the program as well on the relations between components [19].

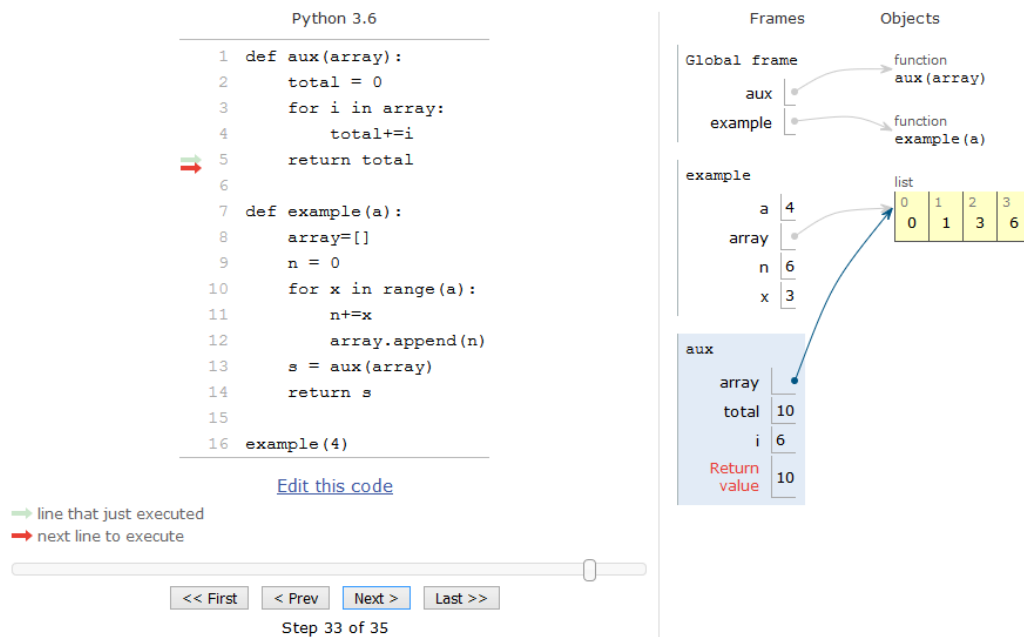
Software visualization is the process of giving a visual representation of a program. It takes the information that is presented in the source code and converts it to a graphical representation with the goal of improving the comprehension of that program. Although a visual image can help to understand a program, it's not that simple since a basic program can have multiple representations. The best one will depend on the task to be performed and what the developer wants to know. This also means that each representation has to be thought for specific tasks and a good portray of the system is imperative for it to be helpful. *Simply repackaging massive textual information into a massive graphical representation is not helpful*, so a linear translation is not ideal since *experts want to see software visualised in context – not just what the code does, but what it means* [14]. The visual representation of a program can be focused in some aspects of the program and it can have different levels of abstraction. This can also solve scalability problems because the visualization of big programs at once it will not be a good help. So, the visualization tool must allow an high level of interactivity to go one step at a time, filter the information, to establish the data that is wanted, to define the level of abstraction and zoom facilities [8]. Graphically these representations can be based on diagrams, maps, icons, graphs and specific drawings. Some examples are the Nassi-Shneiderman diagram used to represent a control-flow of a program [12]; Space-filling (like tree maps or sunbursts) allows to visualize in a very compressed way code metrics and statistics [5]; Graphs are the most common representation and consists of nodes and arcs where the nodes represent blocks of code and the arcs its flow [10].

In terms of functionality, there are two types of visualizations: static and dynamic [10]. A static representation can be seen as a simple image of the program, it doesn't change over time and it's based on static information. Dynamic representations shows the information that changes as the program is executed. A good example of an animated visualization tool is Python Tutor. Python Tutor is an animation web tool that intends to help new programmers to understand programs and currently supports some of the most popular languages at the moment like Python, Java, C, C++, JavaScript, TypeScript and Ruby [15].

An interactive step by step presentation that shows what is really happening behind every line of code during an execution instance (dynamic visualization) is the main feature of Python Tutor. This allows the user to keep up with the modifications of values that each variable suffers as each line of code is executed. The user has total control of this presentation as he can choose if he wants to go to the next or previous step. Global variables are kept in a global frame and then local variables just appear during the function execution.

Tools that produce static visualizations of source code are more usually found. AgileJ StructuredViews [1] is a plugin for eclipse that generates UML class diagrams from the source code. Since UML is a language used to structure software projects that is easy to understand, it can be used to explain to the developers and even to the client. Moreover it creates an easy to read and updated representation of the code. Sourcetrail [3] is a tool that can be connected to an IDE or a text editor that displays an interactive dependency graph and a search bar to search functions, classes or variables. Its interactivity allows the user to see both an overview of the program and the dependencies between classes, methods and variables. CodeSonar [2] offers a visualization software that shows an interactive call

6:4 Graph-Based Visualization Builder



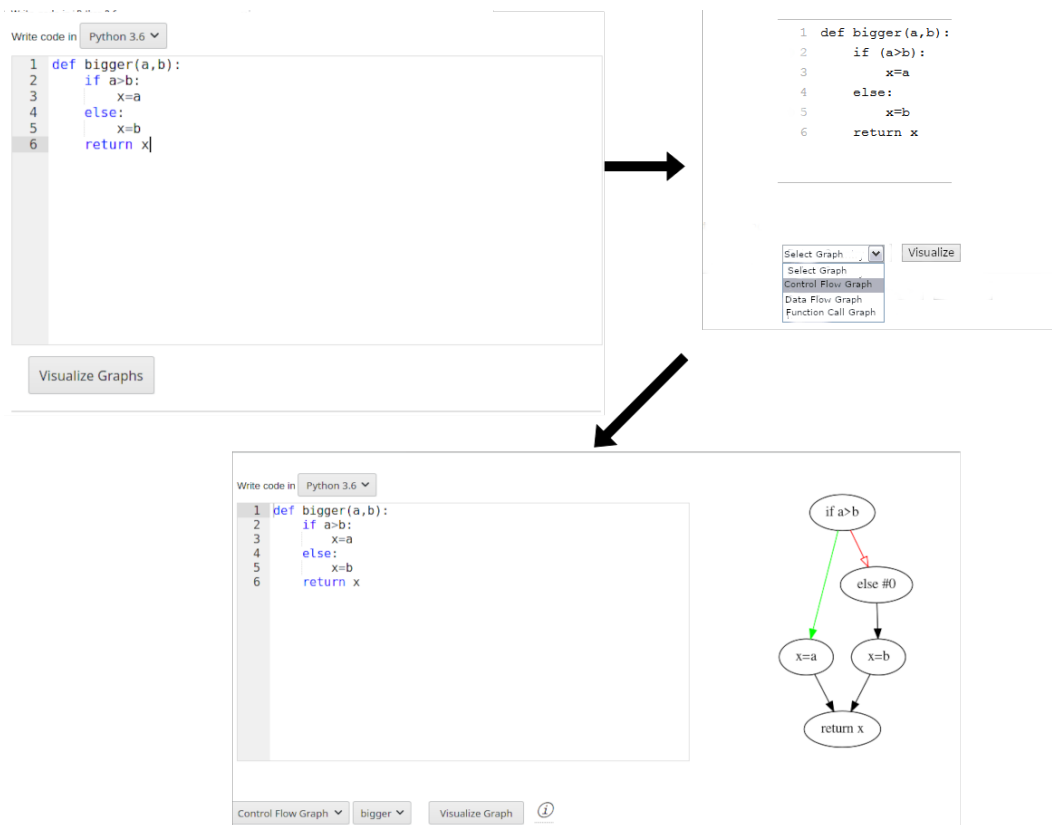
■ **Figure 1** Python Tutor visualization.

graph where it is possible to see the directories, files and functions rearranged hierarchically. It allows the user to choose which metrics he wants to be displayed in the graph as well compare functions/files based on that metrics.

3 Improving Python Tutor with graph-based visualizations

The main idea is to improve the platform Python Tutor by incorporating new features to it. The objective is an interactive tool where the user can select what kind of graph he wants to see in order of being able to analyze visually the flow of code or data dependencies in a part of his code which execution is being animated. Three types of graphs are proposed:

- **Control Flow Graph (CFG):** It is a representation of all paths (sequences of instructions) that might be traversed during the execution of a given program; each node represents a basic block of code (a sequence of instructions without alternatives, with just one input and one output). By observing a control flow graph we can see to where the values go and from where they came. It is a very useful graph to realize the dependencies that exist in the program and which statements are influenced by others. For instance, a statement B is control dependent on a statement A if B execution depends of the A outcome [23].
- **Data Flow Graph (DFG):** In this graph there are two types of nodes where one represents the values/variables and other the operators [21]. This graph allows the user to observe all operations that any variable suffers in its lifetime and in this way understand the existing data dependencies. For instance, it is said that B is data dependent on A when A value is used to compute B. This class of graphs is usually more difficult to understand because they tend to represent a lot of information. Besides that, it is not so linear to read as a CFG for example.
- **Function Call Graph (FCG):** This class of graphs displays the relationships between a function and the functions it calls [9]. On one hand, FCG is useful to understand the functions necessary to execute a function, and this is crucial to local errors and change points. On the other hand, it allows the user to identify the most used functions and

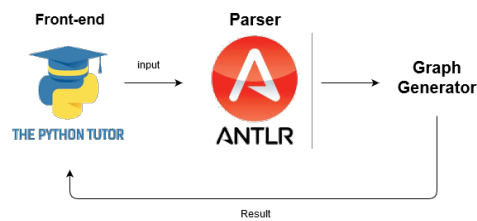


■ **Figure 2** System Workflow.

the ones that are not being used at all. This information is extracted from the program static analysis and knowing which functions are most called can be useful to improve the program performance, since improving these functions will affect a big part of the program. It is, usually, a very easy to read graph.

These new features were incorporated on Python Tutor adding a new field with a scroll bar to select the graph type and a new button to generate the desired graph. Figure 2 shows a diagram of the desired flow.

The source program to be visualized through the dependency graphs is inserted (typed, or edited) at the home page of Python Tutor in order to reuse as much as possible the existing functionalities. So the input text is sent to the back-end of Python Tutor to be compiled. If the input program is correct and ready to be ran and animated, the front-end will receive the execution trace. Once the input is validated, the system will open a new web page created only to visualize the desired graphs. After the user choose the graph that he wants to see, the input is passed through a new route to the graph generator. The graph generator will create an image with the chosen graph and return it to the front-end. The front-end will display the image on that new page so the user can visualize it and then remove the image in order to avoid an overcrowding of files.



■ **Figure 3** System architecture.

4 Integration Architecture and Developments

The Graph Builder was developed in Python using ANTLR to generate the parser and static analyser for the source programs. It is incorporated into Python Tutor to increase its functionality. The user uses Python Tutor editor to write the input program and then it will be parsed by the syntactic analyser generated by ANTLR to identify the control and data dependencies among program elements and represent them as graphs.

This analyser built by ANTLR from the Python grammar is able to recognize Python programs. Once the information about the input program is gathered the next step is to generate the graphs according to that information. Initially the graphs were built in Python with the help of some libraries like 'igraph'. In a second phase, it was used PyGraphViz to build better quality graphs with zoom features and more appealing for users.

Figure 3 shows a diagram depicting the architecture of the developed system.

As said above, to implement the referred parser it was required to define a Python grammar. In this case, it was used the Python grammar available on ANTLR Github. Depending on the information needed to extract, it was required to make some minor modifications on the original grammar. For instance, splitting one grammar rule into three different ones in order to treat the data more specifically. After defining the grammar, semantic actions were specified in order to collect information and generate graphs. The semantic actions were associated to each grammar rule and were constantly adjusted during the development of the graph builder tool. Every time some piece of input code allowed to detect an unhandled case, semantic actions were updated (added or changed) in one or more grammar rules. The way the information was processed had to be as accurate as possible to faithfully build graphs that deal with a large amount of data. So, the structures had to be constantly updated with new information.

The three graphs presented use, by definition, a level of granularity where each node corresponds to an instruction, which can be difficult to visualize for large programs. On the other hand, Python Tutor itself has a limit of lines of code for the input program and even so, for larger programs, it is possible to zoom in and zoom out of the graphs.

In the next section, the final structures chosen to represent each graph will be presented and explained as well the information they hold.

4.1 Function Call Graph

There are two types of functions that can be called in a program: functions defined by the developer or built-in functions (i.e., functions that are available from Python libraries). A function is recognized by an identifier (its name) followed by parenthesis or, in case of being a function written by the programmer, by the keyword `def` followed by its definition. All the function calls found in a program are gathered in a Python dictionary where the key

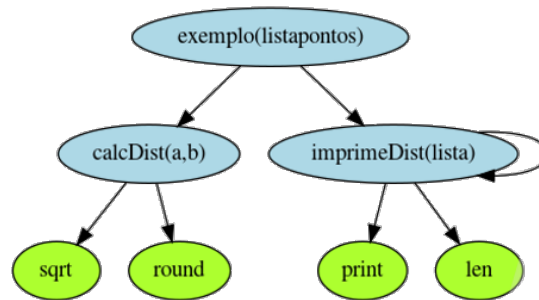

```

def exemplo(listapontos):
    listadist=[]
    for (a,b) in listapontos:
        dist = calcDist(a,b)
        listadist.append(dist)
    imprimeDist(listadist)

def calcDist(a,b):
    dist = sqrt(b**2 + a**2)
    return round(dist)

Def imprimeDist(lista):
    print(lista[0])
    if len(lista)>1:
        imprimeDist(lista[1:])

```



■ **Figure 4** FCG construction.

is the name of the *caller* (the function that calls) and the value is the *callee* (the function called). In Figure 4 it can be seen how the information required to create a Function Call Graph is stored. On the final graph the two types of functions are distinguished by color: blue for the functions created by the user, and green for Python built-in functions.

4.2 Control Flow Graph

The data for the construction of a CFG is stored in a similar way as the one explained for the previous graph, that is represented in a dictionary with an entry for each defined function. As the construction of this graph is not so linear as the previous one, it is required to save more information in the dictionary. So, instead of saving just the code statement, it also saves the type of that statement and the block number where it appears.

Analysing each statement, one can notice that some do not affect the execution flow and others like loop (**for** or **while**) or conditional instructions (**if**, **elif**, **else**) have a strong influence on the flow. Each node of the graph can have one or more edges coming out depending on which type of statement it represents. For example, a node of *loop* or *conditional* type will have two edges coming out of it, one in case of the condition being true and the other in case of being false. A *loop* node will also have at least one more edge coming in from the last statement inside the loop. An *else* node will always be preceded by one node of a non-simple type.

The block number represents the statement level: the first level corresponds to the main, other levels mean that is inside a (possibly nested) block depending on a condition. The body (group of statements that depends on one statement) of non-simple statements will always have a bigger depth level. When the depth level changes from one statement to another, it allows the software to recognize that it will start a new block or that one just ended.

There are other statements that affect the flow of a program like for example the reserved words **break** or **continue**. As can be observed in Figure 6, the CFG also contains coloured arrows in order to be more intuitive for the user to identify which path is followed if the condition present in the node is true (green arrow) or false (red arrow). Besides the color difference, the true path arrowhead is filled unlike the false one that is not. These kind of features can help the user to better read the graph.

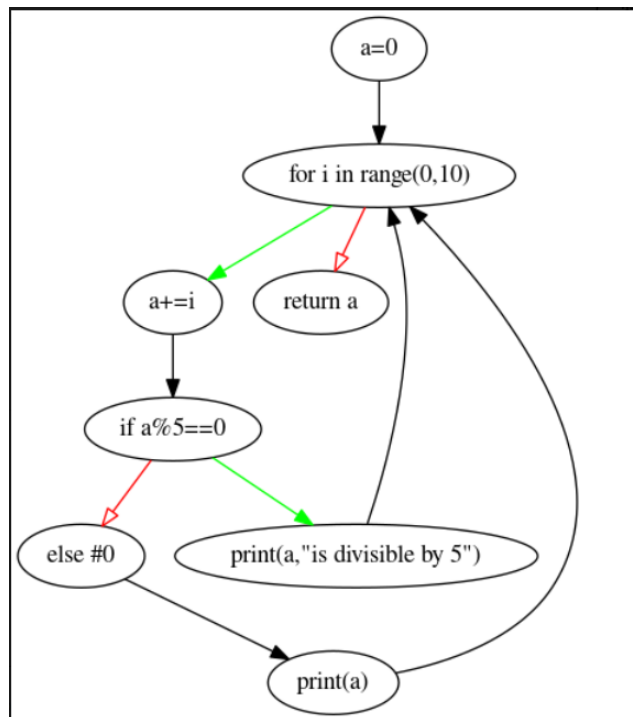
6:8 Graph-Based Visualization Builder

```
def example():
    a = 0
    for i in range(0,10):
        a += i
        if a%5==0:
            print(a, " is divisible by 5")
        else:
            print(a)
    return a
```

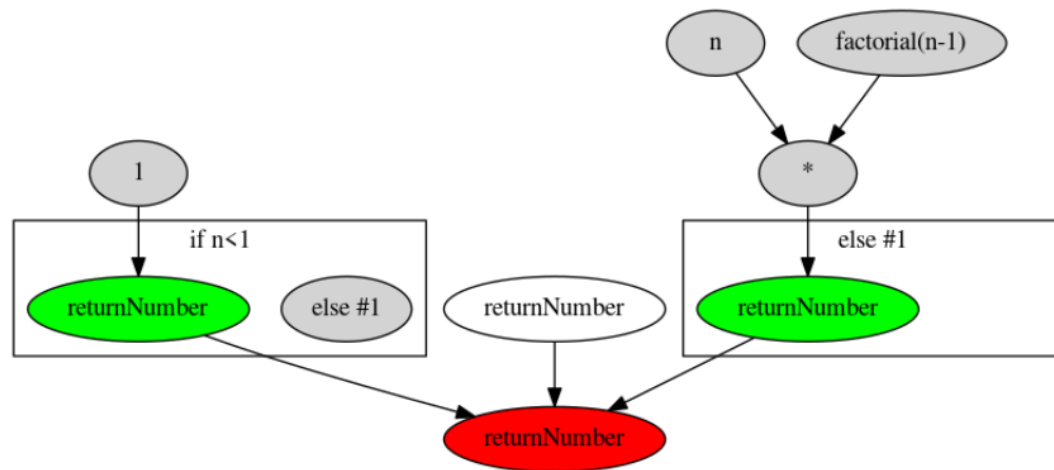


Nº Body	Type	Stmt
0	simple	a = 0
0	loop	for i in range(0,10)
1	simple	a += i
1	if	if a%5==0:
2	simple	print(a, " is divisible by 5")
1	else	else:
2	simple	print(a)
0	simple	return a

■ Figure 5 CFG construction.



■ Figure 6 CFG Example.



■ **Figure 7** DFG construction.

4.3 Data Flow Graph

This sort of graph is not as linear as the previous ones because one variable depends not only on the other variables or function results but also on loops or conditional statements. There are also variables that only exist on a specific scope. Despite these constraints, the information necessary to build DFG is gathered in the same way as the previous ones. In this case, there are three types of tuples in the construction of this graph, one that represents a change of value of a variable (variable name, block number, new value and operations that produced the new value), one that serves to control the loops and conditionals dependencies (depth level, statement, alternative path), and a third one to represent the case when a variable can have multiple values depending on the execution path (all the possible variable values).

As can be seen in Figure 7, it is fundamental to use colors for the nodes in order to get more intuitive drawings. Therefore it is used the color green for the nodes that represent the variables which value changed, and in red the nodes that aggregate all possible values of that variable at a given moment. In addition to use different colors different arrow types are also generated as can be seen in figure 6. The other nodes depict values or operations. This graph also includes boxes surrounding sub-graphs that represent loop blocks (cycles) or conditional blocks aiming at producing clearer and more informative drawings.

5 Python Tutor with Graph Builder

To illustrate the project final outcome, Figure 8 shows a screenshot of Python Tutor integrated with the Graph Builder. In this case it was generated a Control Flow Graph (showed in the right side window) for the input program written in the left side window.

In order to assess the users' satisfaction and the usability of the Graph Builder plugin for Python Tutor, a specific experiment with Python Programmers as testers was designed and conducted. For that purpose, it was created a survey that contained instructions for the testers to follow in order to provide their opinion, answering some questions about the new features. The testers should be able to write Python functions and analyse the resulting graphs. This survey was completed by thirty one participants. The majority of the participants were students or ex-students of a Master's in Informatics Engineering, thus fulfilling the necessary requirements. These participants already have programming knowledge; that fact allows us to get the insights of people familiarized with the world of

6:10 Graph-Based Visualization Builder

The screenshot displays the Python Tutor interface. On the left, a code editor shows Python code for a factorial function and a loop with break/continue statements. On the right, a control flow graph (CFG) visualizes the execution paths. The graph starts with a loop node 'for x in range(10,20)', which branches into an 'if (x==15)' node. From this node, one path leads to a 'break' node and another to an 'if (x%5==0)' node. The 'break' node leads to a 'continue' node, which loops back to the start of the 'for' loop. The 'if (x%5==0)' node leads to a 'print(x)' node, which also loops back to the start of the 'for' loop. Below the code editor, there are dropdown menus for 'Control Flow Graph' and 'breakContinue', and a 'Visualize Graph' button. A tooltip explains that a CFG is a representation of all paths that might be traversed. A privacy policy notice is visible at the bottom.

```
1 def factorial( n ):
2     returnNumber = 0
3     if n <1:
4         returnNumber = 1
5     else:
6         returnNumber = n * factorial( n - 1 )
7     return returnNumber
8
9
10 def breakContinue():
11     for x in range (10,20):
12         if (x == 15):
13             break
14         if (x % 5 == 0):
15             continue
16     print(x)
```

Control Flow Graph | breakContinue | Visualize Graph

A control-flow graph (CFG) is a representation of all paths that might be traversed through a program during its execution.

Privacy Policy: By using Python Tutor, your visualized code, options, user interactions, text chats, and IP address are logged on our server and may be analyzed for research purposes. Nearly all web services collect this basic information from users in their server logs. However, Python Tutor does not collect any personally identifiable information from its users. It uses Google Analytics for website analytics.

■ **Figure 8** Python Tutor with Graph Builder.

programming. The results so far obtained were very positive and increased the confidence on this Python Tutor plugin (the Graph Builder); they also provided a good insight on its relevance in the area of program comprehension. The survey also was helpful to identify some minor bugs in the construction of the graphs; after being fixed, the accuracy and the intuitiveness of the graphs were increased.

5.1 Survey Result Analysis

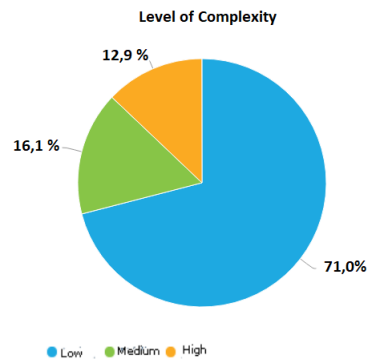
First, it will be presented the results related to the technical side, like the accuracy of the tool and the complexity level of the functions used to test the feature. Then, it will be discussed the opinion of the testers about the features offered by the new Python Tutor plugin. These results, unlike the first ones, will be analysed separately for each graph so that we can identify clearly those that can be more helpful for a deeper program comprehension.

5.1.1 Code Complexity

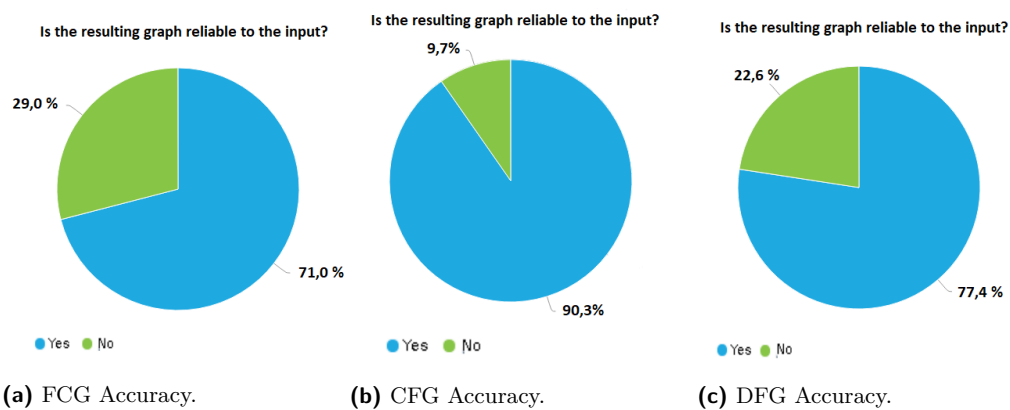
The first question of the survey was related to the complexity of the code the tester wrote. In order to test a major diversity of programs each tester was free to chose the program to use and this question was included to relate the correctness of the resulting graph with its complexity. As it is observable on the pie-chart of Figure 9, the majority of the functions written to test were of low complexity. This is not a problem for the desired assessment because Python Tutor is directed to new programmers and, under these circumstances, the expected inputs are of low to medium complexity. It is important to note that the complexity of the program is subjective to whoever wrote it.

5.1.2 Graphs Correctness

The three graphics in Figure 10 allow to analyse the accuracy for each graph type.



■ **Figure 9** Complexity level of functions used to test the feature.



■ **Figure 10** Accuracy of each type of graph.

The numbers shown in Figure 10 indicate a good percentage of accuracy for each graph. The most surprisingly result being the FCG (see 10a); it was expected to have the highest accuracy but ended up being the one with the least. This may have happened because as this is the simplest graph, the tests performed on this graph were not as intense as the others, not allowing to catch as many exceptions like happened with the other graphs.

On the other hand, it was encouraging to realise how accurate was the CFG (see 10b), and that the DFG got also a good rating (see 10c) mainly keeping in mind the amount of dependencies necessary to show. These results were very satisfying mainly because the bugs reported were corrected which means that this accuracy is now increased.

5.1.3 Graphs and Program Comprehension

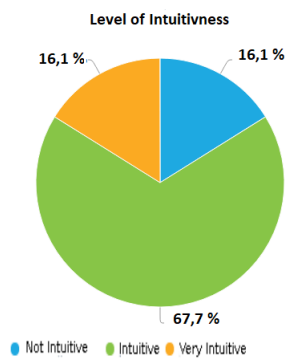
In this subsection we will see whether graphs available can help people on comprehending a program. The survey prepared for the experiment under discussion includes two questions for each graph.

The first question aims at evaluating the intuitiveness of the respective graph, or in other words, perceive whether the graphs are easy to read or not.

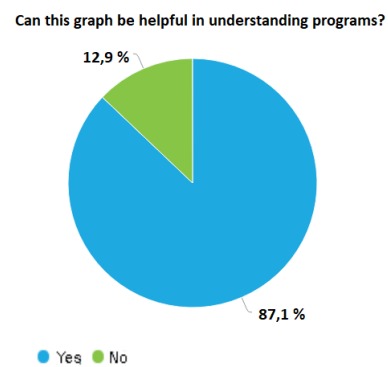
The second question aims at understanding if the tester thinks the graphs can help the comprehension of the program being visualized.

6:12 Graph-Based Visualization Builder

According to the graphic in Figure 11b, 87,1% of the programmers who tested the tool think that FCG can improve a program comprehension what follows the Figure 11a where 83.8% find it intuitive. This is because it is a very simple graph providing a generic overview of the system behavior as it only shows the functions who are called by other functions.



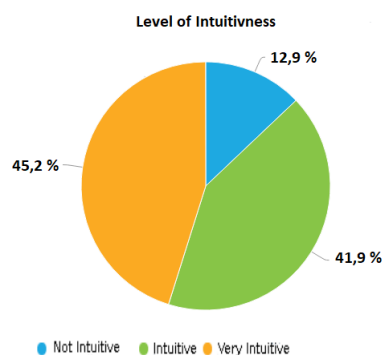
(a) FCG Level of Intuitiveness.



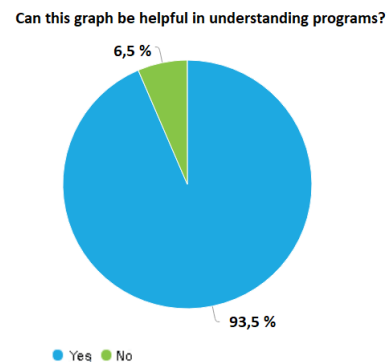
(b) FCG On Program Comprehension.

■ **Figure 11** Perceptions on FCG.

The results shown in the pie graphics of Figure 12 reinforce the preconceived idea that the CFG will be the strength of the new feature. These positive results boost the confidence that this graph can really be helpful on program comprehension.



(a) CFG Level of Intuitiveness.

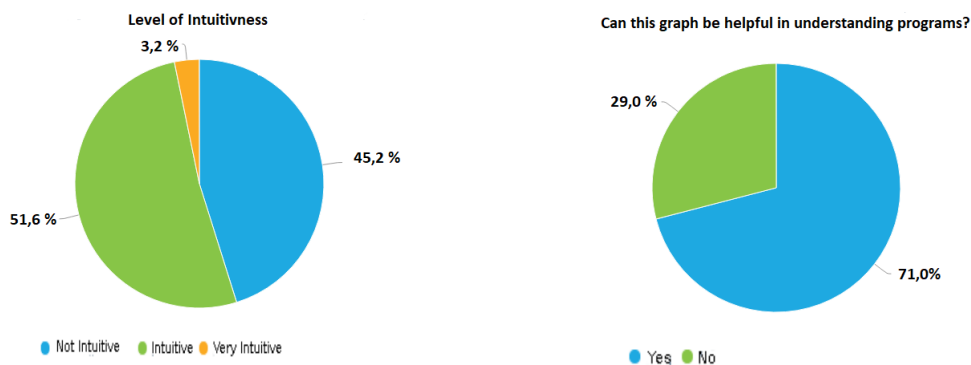


(b) CFG On Program Comprehension.

■ **Figure 12** Perceptions on CFG.

As expected, DFG was considered the least intuitive graph (see Figure 13a for details). As shown in Figure 13b, although the lack of intuitiveness of this graph, a considerable amount of people still thinks that it can help on the comprehension of a program, which means that working out on the improvement of this component is a priority in the near future.

At the end, these results were very positive and increased the confidence on this Python Tutor plugin (Graph Builder) and how it can aid on program comprehension. This survey let us identify some minor bugs made along the construction of the graphs that after being fixed increased their accuracy and intuitiveness.



(a) DFG Level of Intuitiveness.

(b) DFG On Program Comprehension.

■ **Figure 13** Perceptions on DFG.

6 Conclusion

Control Flow Graph (CFG), Data Flow Graph (DFG) and Function Call Graph (FCG) are examples of formal instruments that help to visualize statically some program perspectives that depict the program behavior. Those instruments are considered important for professional programmers to better comprehend the software systems they have to deal with. Taking this statement as proved in area of Program Comprehension, we argued in this paper that they can also be valuable artifacts to help beginner programmers to learn that topic. Moreover, if we can combine the visualization of the referred graphs with program animation tools, we believe that we can motivate and engaged students.

In this context, the paper reported a project aiming at building a tool that transforms a function into its corresponding control and data flow graphs and to integrate it with the program animator tool Python Tutor. At the end, it was proven that the usage of program animation features complemented with dependency graphs visualization tools is feasible and facilitates the comprehension of programs.

The ambition was to make this tool a general one that could help all computer programmers, not only beginners having to learn problem solving and deep their knowledge on a specific programming language, but also professionals carrying out their software maintenance tasks. Even recognizing that scaling up this feature to cope with large, real size, programs is not so effective – because the corresponding graphs are big and confusing, and they require more resources than those provided Python Tutor – we think that the experience succeeded as a support to programming courses since it is covered the statements common to the most used imperative languages.

The process of building this tool encountered obviously some obstacles. The biggest difficulty faced was related to the Python Grammar used that does not cover all the real program situations causing many compiling exceptions that frequently arose. Other challenge was the integration with Python Tutor as this platform resorted to some outdated libraries. However, all these troubles have been overcome ending up with a functional tool to help understanding the algorithm and data structures implemented by small programs that students have to study following an easy, attractive and fast learning process.

The design and conduction of new and more complete experiments involving a larger number of students with different ages and in different programming courses, is the most urgent task that needs to be performed to complete the project. It is important to show that it is an effective way to help programming students. The developed tool (Graph-Builder) is available at <https://graph-builder.di.uminho.pt/visualize.html>

As future work, the first proposal is to implement the same control-flow and data-flow graphs for the other languages presented in the Python Tutor like Java and C. It is expected to be easier to implement it now as the new languages will only need the construction of a grammar for each language as the graph generator can be reused and the integration with Python Tutor would only need a few adaptations. The other direction for the future of this Python Tutor add-on is to improve the aspect of the DFG as the testers inquired said that it is not very intuitive and easy to comprehend.

References

- 1 Agilej structurereviews. <http://www.agilej.com/>.
- 2 Codesonar. <https://www.grammatech.com/products/source-code-analysis>.
- 3 Sourcetrail. <https://www.sourcetrail.com/>.
- 4 Luís Alves, Dušan Gajic, Pedro Rangel Henriques, Vladimir Ivancevic, Vladimir Ivkovic, Maksim Lalic, Ivan Lukovic, Maria João Varanda Pereira, Srđan Popov, and Paula Correia Tavares. C Tutor usage in relation to student achievement and progress: A study of introductory programming courses in Portugal and Serbia. *Computer Applications in Engineering Education*, n/a(n/a), 2020. doi:10.1002/cae.22278.
- 5 Marla Baker and Stephen Eick. Space-filling software visualization. *Journal of Visual Languages & Computing*, 6:119–133, June 1995. doi:10.1006/jvlc.1995.1007.
- 6 Ruven Brooks. Towards a theory of the comprehension of computer programs. *International Journal of Man-Machine Studies*, 18(6):543–554, 1983. doi:10.1016/S0020-7373(83)80031-5.
- 7 Luis M. Gómez-Henríquez. Software visualization: An overview, 2001.
- 8 Martin Hadley. 3 benefits of interactive visualization, January 2018.
- 9 Ben Holland. Call graph construction algorithms explained, March 2016.
- 10 François Lemieux and Martin Salois. Visualization techniques for program comprehension - a literature review. In *SoMeT*, 2006.
- 11 A. Mayrhauser and A. Marie Vans. From program comprehension to tool requirements for an industrial environment. In *[1993] IEEE Second Workshop on Program Comprehension*, pages 78–86, August 1993. doi:10.1109/WPC.1993.263903.
- 12 Ike Nassi and B. Shneiderman. Flowchart techniques for structured programming. *ACM SIGPLAN Notices*, 8:12–26, August 1973. doi:10.1145/953349.953350.
- 13 Michael O'Brien, Jim Buckley, and Teresa Shaft. Expectation-based, inference-based, and bottom-up software comprehension. *Journal of Software Maintenance*, 16:427–447, November 2004. doi:10.1002/smr.307.
- 14 Marian Petre. Mental imagery, visualisation tools and team work. In *Proceedings of the Second Program Visualisation Workshop*, pages 3–14, January 2002.
- 15 Ben Putano. A look at 5 of the most popular programming languages of 2019. <https://stackify.com/popular-programming-languages-2019/>. Accessed: 26-9-2019.
- 16 Gerard Rambally. The influence of color on program readability and comprehensibility. *ACM Sigcse Bulletin*, 18(1):173–181, February 1986. doi:10.1145/5600.5702.
- 17 Teresa M. Shaft and Iris Vessey. The relevance of application domain knowledge: Characterizing the computer program comprehension process. *J. Manage. Inf. Syst.*, 15(1):51–78, 1998. doi:10.1080/07421222.1998.11518196.
- 18 Ben Shneiderman, Richard Mayer, Don McKay, and Peter Heller. Experimental investigations of the utility of detailed flowcharts in programming. *Commun. ACM*, 20:373–381, June 1977. doi:10.1145/359605.359610.
- 19 J. Siegmund. Program comprehension: Past, present, and future. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 5, pages 13–20, March 2016. doi:10.1109/SANER.2016.35.
- 20 E. Soloway and K. Ehrlich. Empirical studies of programming knowledge. *IEEE Transactions on Software Engineering*, SE-10(5):595–609, September 1984. doi:10.1109/TSE.1984.5010283.

- 21 Marilyn Wolf. *Computers As Components, Third Edition: Principles of Embedded Computing System Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2012.
- 22 Mami Yoshida. Think-aloud protocols and type of reading task: The issue of reactivity in l2 reading research. In *Selected Proceedings of the 2007 Second Language Research Forum*, January 2008.
- 23 Andreas Zeller. Chapter 7 - deducing errors. In Andreas Zeller, editor, *Why Programs Fail (Second Edition)*, pages 147–173. Morgan Kaufmann, Boston, second edition edition, 2009. doi:10.1016/B978-0-12-374515-6.00007-1.

Matching User Interfaces to Assess Simple Web Applications

Marco Primo  

Faculty of Sciences, University of Porto, Portugal

José Paulo Leal¹   

Faculty of Sciences, University of Porto, Portugal

CRACS – INESC, Portugal

Abstract

This paper presents ongoing research aiming at the automatic assessment of simple web applications, like those used in introductory web technologies courses. The distinctive feature of the proposed approach is a web interface matching procedure. This matching procedure verifies if the web interface being assessed corresponds to that of a reference application; otherwise, provides detailed feedback on the detected differences. Since web interfaces are event-driven, this matching is instrumental to assess the functionality. After mapping web interface elements from two applications, these can be targeted with events and property changes can be compared. This paper details the proposed matching algorithm and the current state of its implementation. It also discusses future work to embed this approach in a web environment for solving web application exercises with automatic assessment.

2012 ACM Subject Classification Information systems → Web interfaces; Applied computing → Computer-managed instruction

Keywords and phrases automatic assessment, web interfaces, learning environments

Digital Object Identifier 10.4230/OASICS.ICPEEC.2021.7

Category Short Paper

Funding *José Paulo Leal*: 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

Automatic assessment of programming exercises is nowadays a standard practice in introductory programming courses [2]. It usually boils down to a black-box approach, where the assessed program is fed with test data, and the resulting output is compared with the output produced by a reference solution, for the same data. This approach is programming language agnostic and can be easily adapted to a wide range of settings. However, one of its shortcomings is the inability to deal with graphical user interfaces, in particular with web interfaces.

Graphical user interfaces in general, and web interfaces in particular, rely on user interaction for input rather than on data streams. On these applications, data is received as events that are targeted to different elements, without a predefined order. Moreover, web interfaces rely on the articulation of more than a single language – namely HTML, CSS, and JavaScript. These two characteristics prevent the use of simple black-box techniques, typically used for automatic assessment in algorithms and data structure courses.

To overcome this problem we propose a different approach to assess exercises in introductory web technologies courses. These exercises usually involve coding a simple web interface with a small number of widgets, such as buttons, editing fields, and selectors. Although

¹ corresponding author



7:2 Matching User Interfaces to Assess Simple Web Applications

simple, these web interfaces can be implemented in different ways. For instance, a button may be coded as either a `<button>` or `<div>` HTML element. To position these elements, different CSS techniques can be used, such as floating elements or *flexbox*, to name a couple. Last but not least, the exact position and dimension of these elements are immaterial, at least from a pedagogical perspective, provided they preserve geometric relationships (e.g. left-right, center-aligned).

The core of the proposed approach is the matching between two web interfaces: that of the application being assessed and that of a reference solution. The matching procedure is flexible enough to recognize equivalent elements that may have different types, slightly different sizes, or be organized using different structures. On a successful matching, the result is a mapping between the elements of the two web interfaces. On an unsuccessful matching, the result is a set of differences between the two web interfaces that may be used to generate feedback for the student. In the latter case, the assessment is complete and the student must first fix the detected issues.

A mapping between the elements assessed and the reference web interfaces are the basis for testing the web interface features. This stage is based on a sort of unit testing, where data is injected into certain elements as events, and assertions are made on the changes of properties in other elements. The tests that pass on the reference application are then applied to the application under assessment, after replacing the elements using the mapping. At this stage, failed assertions are also reported to the student as feedback, until the exercise is completely solved.

The remainder of this paper is organized as follows. Section 2 presents a literature survey on approaches to compare user interfaces. Section 3 details the proposed algorithm for matching web interfaces based on simple examples, and provides a few implementation details. Finally, Section 4 concludes with future work regarding the implementation of a web environment for solving web application exercises, capable of automatic assessment and feedback.

2 State of the art

The automatic evaluation of programming exercises has become a common practice in introductory programming courses, due to the growing number of new students [9, 10], which renders the manual assessment of programming exercises infeasible or demanding a large number of resources [9].

The literature describes several automatic assessment environments [1], however, none of these tackles efficiently the evaluation of graphical interfaces for educational purposes [1]. These environments help students by promoting feedback that steepens the learning curve [5]. Although the feedback provided by automatic assessment systems is not fully effective as an isolated practice, it becomes an important tool when combined with the teacher's feedback and its use is nowadays a standard in programming courses [10]. Automatic assessments can be divided into two broad categories: static or dynamic [2, 11]. The former analyses the code itself while the latter executes it and analyses its side effects.

To compare the two web pages graphically, different approaches can be taken. One of these approaches is structural comparison, as described in the article [15], this strategy consists of comparing the trees resulting from the two HTML documents. Despite being a possibility, this strategy may not be a solution when it comes to building an automatic evaluation system for web interfaces, given the fact that trees can be the same graphically and structurally different.

Computer vision is another approach that can be used to perform tests on graphical interfaces [3], this approach is based on artificial intelligence models. A construction of an algorithm capable of classifying images and composed by several steps, the main ones being the definition of an adequate classification system, selection of training, pre-processing of images, extraction of characteristics, selection of appropriate approach classifications, post-processing classification and precision evaluation [6]. Due to the inherent complexity of this class of algorithms, the usage of this algorithm can be prohibitive even for comparing simple web interfaces.

To the best of the authors' knowledge, no previous attempts were made to match web interfaces to assess programming exercises. The systems described in the literature to visually compare web pages have very different goals – such as detecting phishing sites [8, 4, 12] –, and are more focused on spotting small differences than ignoring them.

3 Web interface matching

The proposed approach to web interface matching is independent of their internal structures. This objective is achieved by making use of the element's properties of each page and the spatial relationships between them. Moreover, it was designed to produce incremental feedback that may help students overcoming their difficulties.

As in program assessment in general, web interface assessment compares a program against a model. It is a dynamic assessment since it compares the side effects of both programs' executions; in this case, the web interfaces they produce. To access one against the other, these web interfaces need to be related. This relation is created by using a comparison algorithm between web interfaces.

The approach we used to create an algorithm to relate web interfaces consists of mapping among leaf elements. Leaf elements, also known as *widgets* in user interface frameworks, are the elements used as leaves in a web interface tree structure, namely the leaves *DOM* (Document Object Model) [14] object structure. Non-leaf elements are less relevant since different structures, used for controlling the position of leaf elements, may lead to graphically similar web interfaces. Moreover, leaf elements of different types may have a similar look and feel. For instance, a button may be implemented both as a `<button>` or `<div>` element.

The proposed algorithm for comparing web interfaces operates of sets of leaf elements in 4 consecutive phases:

Initial sets creation: Creates two sets of leaf elements from each web interface, including their properties. This data is obtained using the *JavaScript* binding of the *DOM* interface.

Attribute refinement: In both sets, for each element, the properties that are not relevant to the algorithm are removed. The resulting sets are lists of *JSON* [7] objects containing properties, preserving some of the original properties, and computing new ones. Currently, the preserved properties are `name`, `id`, `tag`, `internal text`, and `src`. Two of the original properties – `offsetTop` and `offsetLeft` – are used to compute new properties to capture the spatial relations among elements.

Element mapping: The result of this phase is a set of pairs, one from each leaf element set. The rules that pair elements ensure that the most similar are created first. This greedy approach is used to ensure that a good mapping between all elements is efficiently created.

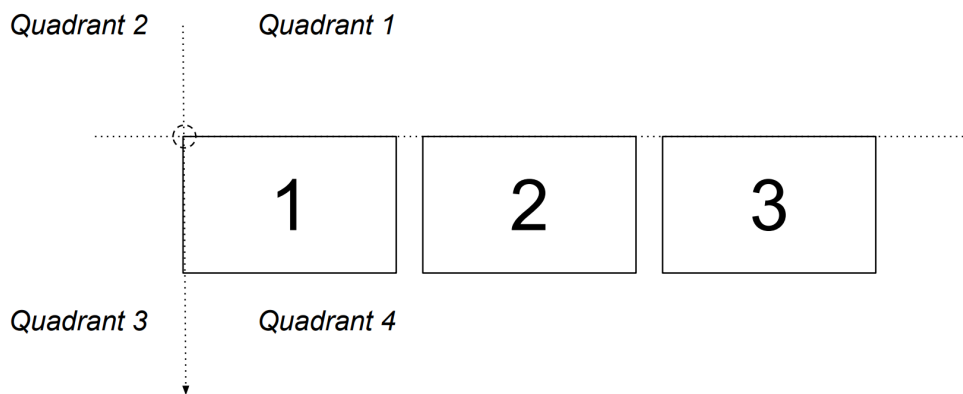
Feedback generation: The set of pairs is the basis for feedback generation. It should be noted that this set can only be created if initial sets have the same cardinality. Otherwise, feedback must identify the elements from the tested web interface that are missing, or

7:4 Matching User Interfaces to Assess Simple Web Applications

that could not be mapped. Even with a complete set of pairs, some of them may be partial matches. That is, some of their relevant properties may be different. Using this data higher granularity feedback is generated.

The similarity between elements of a pair is computed as a score obtained by comparing their properties. Due to some properties being more decisive than others, this score is a weighted sum.

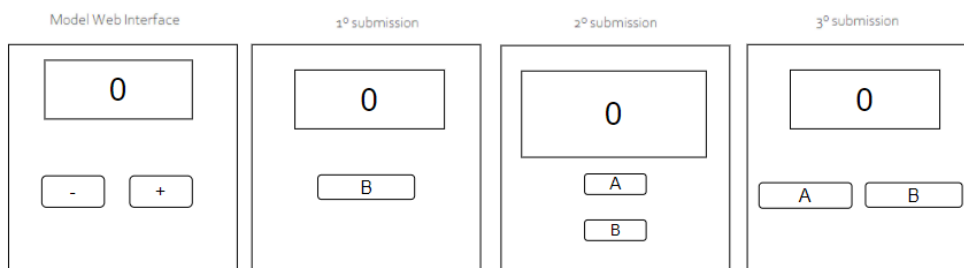
The comparison of properties governing the elements' positions and sizes is the most challenging part of the proposed approach. This is done by considering relative spatial relationships between the elements of the same web interface. These relations dictate in which geometric quadrant the element is concerning the other.



■ **Figure 1** Example of quadrant based spatial relationships.

Consider a web interface with 3 non-overlapping leaf elements, aligned horizontally, with IDs 1, 2, and 3, as shown in Figure 1. Using the upper left corner as referential, following the convention in windowing systems, one can define relative spatial relations with the other elements. For element 1, elements 2 and 3 are in quadrant 4, as they are mostly to the right and below the upper left corner of element 1.

The attribute refinement phase adds new properties to each element counting the number of other elements (the element itself is not counted) lying in each of the 4 quadrants, using its upper left corner and referential. In the example of Figure 1: element 1 has 2 elements in the 4th quadrant; element 2 has 1 element in the 3rd quadrant and 1 in the 4th, and element 3 has 2 elements in the 3rd quadrant. The unmentioned quadrants of each element count 0 elements.



■ **Figure 2** Example of incremental assessment and feedback.

Figure 2 illustrates successive steps in solving a web interface exercise made by the student until having it accepted, guided by the algorithm's feedback. The model application is a simple counter, with a label that displays the current count and two buttons, one to increase the count and another to decrease it. When the buttons are clicked, the displayed number is changed accordingly.

After the 1st submission, the algorithm cannot map all the leaf elements. However, it can identify a missing button and that information is reported as feedback to the student. Using that feedback, the student adds a button to the web interface. After the 2st submission, all elements of the student's web interface are mapped to those on the model web interface. However, they differ both on relative spatial relation and the buttons' internal text. To avoid overloading the student with excessive feedback [13] the information on non spatial errors is omitted from the feedback.

Subsequently, on a 3rd submission, all elements continue to be identified. However, the buttons' internal text still differs and that information is given as feedback. In the final submission, not shown in the figure, the student replaces "A" and "B", by "+" and "-", respectively, and the submission is accepted. There are still differences between the model and the submitted program but they are considered close enough for the intended purpose.

It should be noted that the algorithm can be configured to consider more (or less) properties as relevant. For instance, the size properties could have been considered relevant and the last submission would not be accepted, since the button's size is larger than those of the model web interface.

4 Future work

In our ongoing work, we plan to use the algorithm presented in Section 3 as the cornerstone of a web interface assessment environment. This environment will assess both the appearance and the interactivity of the web interfaces developed as exercises. The mapping algorithm maps elements in both web interfaces. Thus, functions that check invariants on the model interface, as a sort of unit tests, can be applied to the student's submission.

Consider again the model interface shown in Figure 2. A unit test can: 1) start by obtaining the label's initial value; 2) then send a click event to the increase button; 3) and, finally, check that label's value is incremented. The same unit test can be applied to a similar web interface, provided that the element references are replaced by the corresponding elements on the web interface being tested.

As the mapping between the web interfaces is done relatively, this means that measures and absolute positions become irrelevant, a web interface can be assessed as correct if it maintains the expected relative positions among the elements. With an evaluation that does not force the tested web interface to follow the same structure of the model, we expect to increase the fairness of the assessment environment.

References

- 1 Dr SM Afroz, N Elezabeth Rani, and N Indira Priyadarshini. Web application—a study on comparing software testing tools. *International Journal of Computer Science and Telecommunications*, 2(3):1–6, 2011.
- 2 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.
- 3 Tsung-Hsiang Chang, Tom Yeh, and Robert C. Miller. Gui testing using computer vision. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, page 1535–1544, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1753326.1753555.

- 4 Iginio Corona, Battista Biggio, Matteo Contini, Luca Piras, Roberto Corda, Mauro Mereu, Guido Mureddu, Davide Ariu, and Fabio Roli. Deltaphish: Detecting phishing webpages in compromised websites. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *Computer Security – ESORICS 2017*, pages 370–388, Cham, 2017. Springer International Publishing.
- 5 H. Fangohr, Neil S. O’Brien, A. Prabhakar, and Arti Kashyap. Teaching python programming with automatic assessment and feedback provision. *ArXiv*, abs/1509.03556, 2015.
- 6 D. Lu and Q. Weng. A survey of image classification methods and techniques for improving classification performance. *International Journal of Remote Sensing*, 28(5):823–870, 2007. doi:10.1080/01431160600746456.
- 7 Felipe Pezoa, Juan L. Reutter, Fernando Suarez, Martín Ugarte, and Domagoj Vrgoč. Foundations of json schema. In *Proceedings of the 25th International Conference on World Wide Web, WWW ’16*, page 263–273, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee. doi:10.1145/2872427.2883029.
- 8 S. Roopak and Tony Thomas. A novel phishing page detection mechanism using html source code comparison and cosine similarity. In *2014 Fourth International Conference on Advances in Computing and Communications*, pages 167–170, 2014. doi:10.1109/ICACC.2014.47.
- 9 Riku Saikkonen, Lauri Malmi, and Ari Korhonen. Fully automatic assessment of programming exercises. *SIGCSE Bull.*, 33(3):133–136, 2001. doi:10.1145/507758.377666.
- 10 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, February 2018. doi:10.1002/cae.21974.
- 11 M. Varga and M. Kvassay. Unit testing in data structures graphical learning environment. In *2019 17th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pages 797–804, 2019. doi:10.1109/ICETA48886.2019.9040071.
- 12 Gaurav Varshney, Manoj Misra, and Pradeep K Atrey. A survey and classification of web phishing detection schemes. *Security and Communication Networks*, 9(18):6266–6284, 2016.
- 13 Eric M Wilcox, J William Atwood, Margaret M Burnett, Jonathan J Cadiz, and Curtis R Cook. Does continuous visual feedback aid debugging in direct-manipulation programming systems? In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, pages 258–265, 1997.
- 14 Lauren Wood, Arnaud Le Hors, Vidur Apparao, Steve Byrne, Mike Champion, Scott Isaacs, Ian Jacobs, Gavin Nicol, Jonathan Robie, Robert Sutor, et al. Document object model (dom) level 1 specification. *W3C recommendation*, 1, 1998.
- 15 Jiří Štěpánek and Monika Šimková. Comparing web pages in terms of inner structure. *Procedia - Social and Behavioral Sciences*, 83:458–462, 2013. 2nd World Conference on Educational Technology Research. doi:10.1016/j.sbspro.2013.06.090.

Active Methodologies in Incoming Programming Classes

João Paulo Aires ✉

Departamento Acadêmico de Computação (DAINF),
Federal University of Technology of Paraná (UTFPR), Brasil

Simone Bello Kaminski Aires ✉

Departamento Acadêmico de Computação (DAINF),
Federal University of Technology of Paraná (UTFPR), Brasil

Maria João Varanda Pereira ✉ 

Research Centre in Digitalization and Intelligent Robotics (CeDRI),
Polytechnic Institute of Bragança, Portugal

Luís M. Alves¹ ✉ 

Research Centre in Digitalization and Intelligent Robotics (CeDRI),
Polytechnic Institute of Bragança, Portugal

Abstract

Innovative approaches in teaching programming have been required to improve the success of incoming programming students. This work presents the initial results of a teaching strategy implemented in the Algorithms subject of a Computer Science course. Ninety-five students, enrolled in this subject during the first semester of the course, participated in the research. The reported activity is related with active methodologies of teaching and Problem-Based Learning, being developed on the first day of class in groups of up to five students. The activity was based in two actions: 1) answering a questionnaire associating computing elements to daily life routines; and, 2) even without programming concepts knowledge, develop a smartphone application. Each group received a questionnaire containing 19 questions, divided into four blocks. What can be perceived with the accomplishment of this work, was the enthusiasm, motivation and engagement of the students who, even being unknown from each other, organized themselves in the groups and researched the necessary strategies to complete the challenge. The teacher acted as an advisor in the teaching process, conducting the experiment in order to lead students to find the solution.

2012 ACM Subject Classification Social and professional topics → Computer science education; Software and its engineering → Imperative languages

Keywords and phrases Teaching Programming, Active Methodologies, Learning Innovation

Digital Object Identifier 10.4230/OASICS.ICPEC.2021.8

Category Short Paper

Funding This work has been supported by FCT - Fundação para a Ciência e Tecnologia within the Projects Scopes: UIDB/05757/2020 and UIDB/00319/2020.

1 Introduction

Nowadays there are countless advances that strengthen the educational environment [2] and, among them, two factors stand out: science and technology. Complementarily, Aires and Pilatti [2] emphasize that it is essential at all levels of education (from basic education to postgraduate), to have two fundamental elements: 1) people enthusiastic in the transformation of educational processes; 2) school/university management combined with community

¹ to mark corresponding author



interests. Studies indicate that students' demotivation and the consequent failure rate in undergraduate courses is associated with factors inside and outside the classroom [11, 12], highlighting: the efficiency of the teaching methodology used by the teacher; the immaturity of many students when entering higher education; and, the previous knowledge brought by the student and that can favor a meaningful learning [11, 12, 22].

Thus, for a successful learning, it is necessary that both the student and the teacher apply differentiated teaching methods and that they deem appropriate, in order to promote the absorption of content in an appropriate way [1]. In view of this scenario, with the objective of promoting student engagement and motivation, and consequently expanding the use of the contents worked on, we sought to apply a teaching and learning methodology based on problems. This methodology was used on the first day of class in the subject of Algorithms in an undergraduate course in Computing.

The subject of Algorithms, which has a workload of 90 hours, belongs to the first semester of the course and aims to introduce the student to the universe of programming and the problem solving. The syllabus of this subject contains the introduction to variables, data types, conditional and cyclic structures and manipulation of homogeneous data structures (unidimensional and multidimensional). Since it is a discipline that has historically a high failure rate and student dropout, it is necessary to use teaching strategies that facilitate the learning of content, the motivation to attend the course and school success.

Regarding the activity described in this article, it should be noted that, because it is carried out on the first day, it does not contemplate or intend to contemplate all the basic programming contents, but will only serve as a motivation for the art of programming. In the initial approach of the task, the initial contents of the discipline are identified/explored, such as: assignment, input and output of data, types of data, variables, memory allocation. Problem solving is also addressed using conditional and repetitive structures, control variables and modularization. But all of this is addressed as the need arises to be discussed in order to solve the proposed problem.

The experiment described in this paper shows that an innovative approach can make the difference in terms of motivation level and dropout reduction in the first programming classes. It enhanced the integration of the students and the cooperation between classmates from the very first day of classes, through the formation of groups and discussion about the problem presented by the teacher. Such a problem makes a connection between the real world (common sense knowledge) and the discipline of initial programming. The main objective of the experiment was to get students to reflect on basic/elementary concepts of computing (such as: operating system, memory, processing, storage, data types, input and output, among others), which are present on their daily life, especially on their smartphones. It is important to say that, to carry out the activity, it was not necessary any previous knowledge about programming.

2 Active methodologies in the teaching and learning process

Active Teaching and Learning Methodologies can be understood as any teaching strategies aimed at promoting the effective participation of students in the organization and continuous construction of their learning in a flexible way, adaptable to each one and leading to meaningful learning [3, 4]. Such active methodologies favor the contact with new experiences for students, whether in interventions promoted by the teacher, or in discussions promoted with colleagues. Through them, the student, in an autonomous way, seeks the necessary elements for the consolidation of knowledge, contributing to his personal and professional life, because, in addition to strengthening his role, he improves the ability to face daily

challenges. Additionally, Mitre et al. [18] argue that active methodologies should always use problematization, in order to challenge and motivate the student, since the problem allows the class to analyze the objectives, reflect on the hypotheses, relate the possible solutions and test/record the findings obtained. Learning through this strategy will allow maximum student involvement, acting actively as protagonist in the professional training process.

Finally, following Moran [19], active methodologies are based on the student as the center of learning, being associated in individual or group activities, participating in learning processes in a collaborative way and through the exchange of experiences.

According to Barseghian [6], Spricigo [21], Farias et al. [9], Mazur [15], Leal et al. [5], Silva, Sales and Castro [20], Larmer and Mergendoller [14], Hanney [13] and Bender [7] there are several strategies to be used in the classroom, especially: expository class dialogue; case study; Team-Based Learning (TBL); Problem-Based Learning (PBL); Project-Based Learning (PBL); peer instruction; flipped classroom; gamification. Studies by Mitre et al. [18], Barseghian[6], Berbel [8] point out that the use of active methodologies enables new experiences for the class, regardless of the content worked on. The student, on his own and knowing the real need for such action, seeks the new, and this initiative contributes greatly to becoming critical and reflective in his professional life, working on autonomy and the ability to face challenges. However, if on the one hand it is up to the student to seek additional concepts to solve a certain problem, on the other hand, the application of these strategies depends, essentially, on an effort by the teacher to reorganize the discipline aimed at given to the students the main role.

3 Methodology

This research work, which is under development, follows an exploratory approach, using a predominantly qualitative analysis of the problem. In relation to technical procedures, a survey was carried out, and the composition of the documentary corpus was based on questionnaires collected from students who participated in the project. The study was applied during two semesters (2019-2 and 2020-1), in the initial classes of the Algorithms discipline (taught to new students). 95 students participated (47 students in 2019 and 48 students in 2020), enrolled in the first period of the mentioned discipline. For the purposes of this study, we used the active Problem-Based Learning (PBL) methodology, which aims to place students facing a real problem, to be solved through the creation of working groups. In order to generate engagement and interest with the content of the discipline, the teacher uses the first class to present a problem related with something that is strongly familiar to all the students: the use of smartphones. The main functionalities of a smartphone are discussed with students, such as capturing photos, making video calls, editing e-mail, accessing social networks. Then a questionnaire is proposed to the students to reflect on internal features/characteristics of the device. In the questionnaire, students are invited to immerse/reflect on their smartphone. Many questions were asked based on the essential settings that are evaluated when someone is looking for a cell phone to buy. According to Felder and Brent [10], the tasks to be distributed to students must be organized in such a way as to be carried out in a short time, so as not to discourage the student from participating. Thus, the questionnaire (which contains 19 questions in total) was divided into four blocks/parts and can be viewed at <http://encurtador.com.br/krTHT>. Such activity was structured in such a way that the questions (however basic they may seem) were related to the world of computing, seeking a correlation with the real world and the initial contents of the programming discipline. Then, a challenge is launched to students: to develop an APP for smartphone. After the initial reflections and discussions, the strategy was composed by the following steps:

8:4 Active Methodologies in Programming

1. Presentation of the problem;
2. Possibility to develop the APP in a group or individually;
3. Without explaining about refined programming concepts, the class was asked to develop a smartphone application;
4. The development would follow some rules and should be carried out through MIT App Inventor [16];
5. The APP should be installed and tested on the smartphone;
6. Within a week, students present the developed applications.

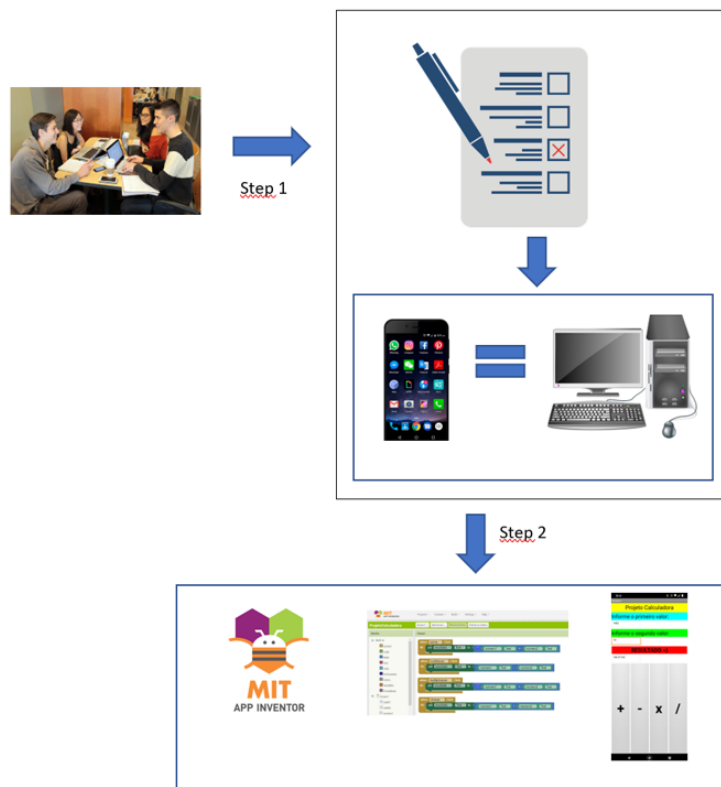
To carry out the proposed activity as a challenge, the tool developed by MIT called App Inventor was used. The MIT App Inventor (available at <https://appinventor.mit.edu/>) is a block-based visual programming environment that allows even beginners to develop full (and fully functional) applications for Android devices. The platform does not require programming in any language, making the implementation of the features very intuitive using a process of *dragging* blocks of commands. As it is a visual environment, as the blocks are inserted (that is, some functionality is implemented) it is possible to visualize the change instantly, through the simulator attached to the platform. At the end of the implementation (and the tests carried out in the simulation), an .apk file can be created so that the application can be installed on the Android device. It is important to highlight that, despite not needing knowledge of programming languages, it is important to understand that the construction of thinking to solve the problem must be created step by step, from beginning to end, just as in an algorithm. Thus, when the teacher explains the general concept of Algorithm (finite sequence of steps, executed in a certain order), the student is already able to understand how the blocks should be organized, so that their application works properly.

Figure 1 shows steps 1 and 2 developed in the proposed methodology. Initially (step 1) the groups of students answer the questionnaire that relates functionalities contained in their smartphones with the functionalities of a computer. After discussing the questions proposed in the questionnaires, students are encouraged to develop an APP for their smartphones (step 2). The images presented in Figure 1 were extracted from the APPs developed by the students.

Figure 2 shows the Team 1 block program for the Calculator problem using App Inventor. As we can see, the pieces fit together forming a puzzle. Students have the possibility to learn programming logic in a playful and intuitive way. Access to a complete code, developed by the groups, is available at <http://encurtador.com.br/dfvV7>.

4 Discussion

The research activities developed in this experiment were based on active teaching and learning methodologies using groups with a maximum of five students. Since the students are new in the Computing Science Course, the research was conducted in order to carefully choose the topics to be worked on, as well as the reasoning sequence (the steps) to obtain the desired learning results. Following Felder and Brent [10], the activity time cannot exceed 20 minutes because after that time the students will lose the focus. In this way, the questions were presented and discussed in four blocks containing four to six questions each. Each block of question is presented to the groups and, afterwards, a discussion is made with the whole class, collecting the answers obtained by the groups. It is important to note that some students participate more as listeners, when they are shy or have little knowledge about the subject. It is part of the teacher's task to form the groups, so that the challenges are developed collaboratively, with the participation of all. However, there is no way to

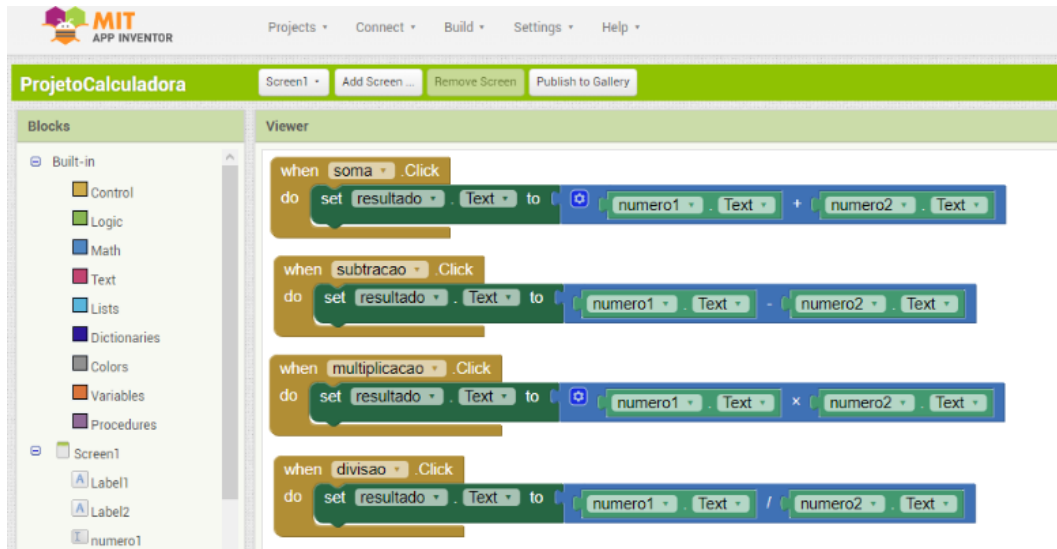


■ **Figure 1** PBL - APP smartphone.

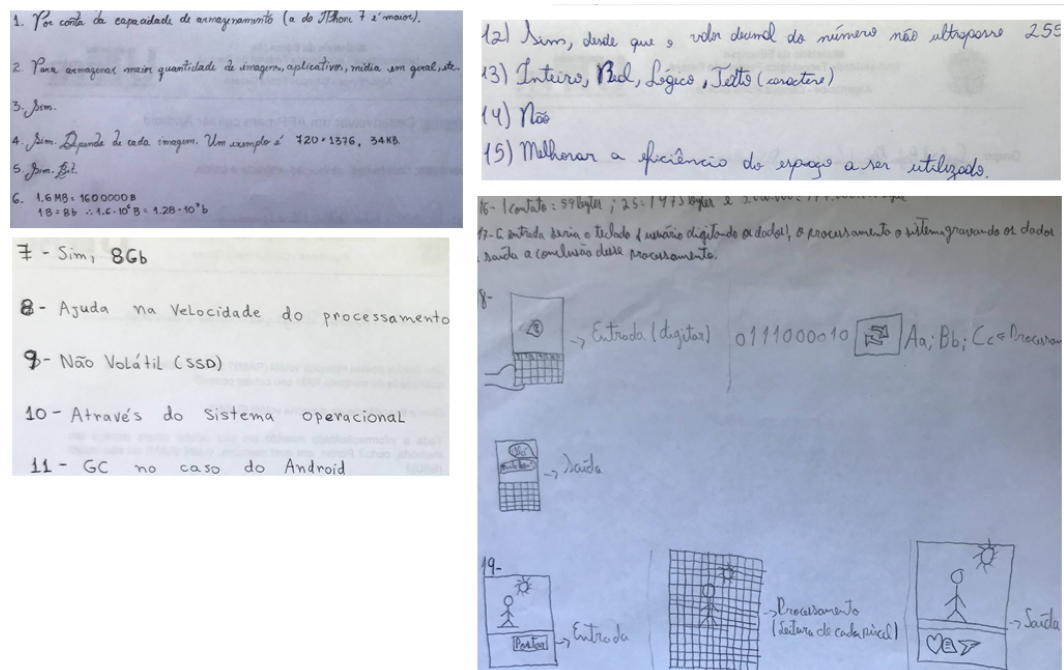
guarantee that this will happen at every stage of the process. For each block of questions, the applied dynamics occurs in two moments: initially, the groups meet, explore and discuss the presented questions and after ten minutes of discussion in the group, the debate expands with the whole class, in which each group presents its answers. At this moment, cooperation between groups was observed and, often, a group complements the answer of the previous one. It is up to the teacher to mediate/organize the answers presented and instigate cooperation between the groups and how the information can be complementary. Whenever necessary, the teacher must explain the content not understood or complete any important information that has not been observed by any group. Figure 3 shows the blocks of responses registered in different groups, taking into account the 19 questions made available in the class activity (responses from other groups, can be viewed at <http://encurtador.com.br/epvLP>). As pointed out in the methodology, the number of activities was dimensioned in order to avoid a low student involvement.

In Figure 3 can be seen that the answers follow the particularities of each group's smartphones. It is also observed that some groups recorded different information for the same question, due to the configuration of the devices of the group members. This was very interesting, as it demonstrates that they perceived the existence of different configurations, favoring them to carry out a more in-depth analysis, at the time of a future acquisition. When the problem is presented to the class, it is observed that students with improved knowledge asked if they can develop the APP using their previous knowledge in programming. On the other hand, students who have no experience about the programming, are apprehensive and scared, thinking about how to solve the task. At this point, it is up to the teacher to

8:6 Active Methodologies in Programming



■ **Figure 2** Team 1 block program for the Calculator problem.



■ **Figure 3** Answers from different groups.



■ **Figure 4** Calculator.

present possibilities to solve the problem. Among them, the MIT App inventor platform (developed by MIT) is explained, in which it is possible to perform the task using only block programming (similar to what happens in Scratch [17]). The teacher's intervention, guiding and conducting the students, does not solve the problem by itself. It works just as a tutoring/orientation task, guiding the students research in order to autonomously reflect and develop the proposed problem. After the questionnaire discussions, students had two days to present the solution to the problem (the APP developed and working on the smartphone). The Figure 4 illustrates some of the application screens developed by the students, proving that all the groups found a solution to the proposed challenge. Note that it was possible to construct a big diversity of interfaces and different ideas had emerged without initially need to know a programming language.

The experience proved to be valid and interesting, due to the involvement level of the students, their integration in the groups, their motivation in the promoted discussions and the developed applications, thus the objective of the activity was accomplish. In addition, it should be noted that the anxiety and concern of students entering the course were overcome by the possibility of getting involved in a work group from the beginning. What at first was something that shocked them, because they imagined they were not able to accomplish it, generated additional motivation and dispute in the class, to see who could finish first. The exchange of the APP developed among students was observed, as well as others implemented by them on the same platform, which was possible with the experience and knowledge acquired in this activity.

In order to verify the understanding of the content worked on and the learning of the initial concepts of computing, an evaluation was carried out with the class using the Moodle Platform. The assessment contained 13 questions (10 of which were multiple choice and 3 descriptive) with the aim of verifying how each student would make the relationship between the problem studied and the concepts inserted in the proposed challenge. Some examples of those questions are:

1. Considering the questions discussed in the activity on cell phones, what questions/subjects brought you new knowledge?
2. In general, would you know how to explain the computer system you use when sending an e-mail? What would be the input(s), processing(s) and output(s)?
3. Who manages the memory space on the cell phone?

4. When we install an APP on our cell phone is it stored in non-volatile memory (SSD)?
 5. Whenever I open an APP (for example: Instagram) it is loaded into the SSD memory?
- The class consisted of 87 students and, of these, 67 answered the questionnaire applied, with an average grade of 7.7. Of the total responses, 56 students (84% of participants) scored higher than the university average (6.0 points) and 11 students (16%) scored lower but six of these ones scored between 5.1 and 5.8 (close to the average pass). Facing and analyzing the results obtained, it is possible to state that the strategy used in the classroom obtained positive results, since the students were able to relate the concepts addressed in the activity with the content necessary for the beginning of the discipline, since a good part of the class (84%) scored higher than the average required for approval.

5 Conclusion

When the students faced the problem to be solved (APP development for smartphone) they reacted with concern saying: *How can I develop an application if I never programmed?, How can I solve that?, How to start?, How to develop?.* However, after research and discussion with colleagues, the initial fear was replaced by the motivation to find the solution, develop the APP, install on the smartphone and see it working (just like any native APP or that they have installed). The differentiated teaching methodology, as pointed out by Fragelli [11] and Tavares [22], provided moments of enthusiasm, involvement in the class and integration between colleagues who had just met. The initial impact and anxiety gave way to motivation for the content worked on, with an important detail: many of the students who participated in this experience, had not even had contact with programming before entering the course. It was noted that the strategy used was well received by the class, which allows advancing in the application of new (and innovative) methodologies so that the contents were better assimilated by students and provides them a meaningful learning. For future work, the knowledge acquired by each student will be individually evaluated through questions related to the questionnaire discussed in group and the APP developed. In this way it is possible to verify the involvement of each student in the activity, assessing the size of the groups and the need to reinforce some unconsolidated content. Furthermore, it is intended to expand cooperation between groups, proposing more complex problems that have dependencies on solutions developed by different groups. The idea is to simulate a work environment, in which the final solution will depend on the results and cooperation between the teams.

References

- 1 João Paulo Aires and Luiz Alberto Pilatti. Aprendizagem significativa por meio do ensino adaptativo. *Revista Espacios*, 37(29), 2016.
- 2 João Paulo Aires and Luiz Alberto Pilatti. Feyerabend promovendo a aprendizagem significativa por intermédio do ensino adaptativo. In *IV Congresso Nacional de Educação (CONEDU 2017)*. Editora Realize, 2017.
- 3 David P. Ausubel. *Aquisição e Retenção de Conhecimentos: Uma Perspectiva Cognitiva*. Plátano-Edições Técnicas, 04-2003 edition, 2003.
- 4 Lilian Bacich and José Moran. *Metodologias Ativas para uma Educação Inovadora*. Penso, Porto Alegre, 10-2017 edition, 2017.
- 5 Lilian Bacich and José Moran. *Revolucionando a sala de aula: como envolver o estudante aplicando técnicas de metodologias ativas de aprendizagem*. SATlas, 1. edição edition, 2017.
- 6 Tina Barseghian. Three trends that define the future of teaching and learning. <https://www.kqed.org/mindshift/7854/three-trends-that-define-the-future-of-teaching-and-learning>. Accessed: 2021-02-09.

- 7 William Bender. *Project-Based Learning: Differentiating Instruction for the 21st Century*. Corwin, 1st edition, 2012.
- 8 Neusi A. N. Berbel. As metodologias ativas e a promoção da autonomia de estudantes. *Ciências Sociais e Humanas*, 32(1):25–40, 2011.
- 9 Pablo Farias, Ana Martin, and CĂn্থia Cristo. Aprendizagem ativa na educação em saúde: Percurso histórico e aplicações. *Revista Brasileira de Educação Médica*, 39(1):143–150, 2015.
- 10 Richard M. Felder and Rebecca Brent. Active learning: An introduction. *ASQ Higher Education Brief*, 2(4), 2009.
- 11 Ricardo R. Fragelli. Trezentos: Aprendizagem colaborativa como uma alternativa ao problema da ansiedade em provas. *Revista Eletrônica Gestão & Saúde Brasília*, 6(2):860–872, 2015.
- 12 Ricardo R. Fragelli and T. B. Fragelli. Three hundred: the human dimension of the method. *Educar em Revista*, 1(63):253–265, 2017.
- 13 Roy Hanney. Doing, being, becoming: a historical appraisal of the modalities of project-based learning. *Teaching in Higher Education*, 23(6):769–783, 2018.
- 14 John Larmer and John Mergendoller. *Setting the Standard for Project Based Learning*. ASCD, Alexandria, VA, USA, 2015.
- 15 Eric Mazur. *Peer Instruction: a revolução da aprendizagem ativa*. Penso, Porto Alegre, 2015.
- 16 MIT. Mit app inventor. <https://appinventor.mit.edu/>. Accessed: 2021-03-08.
- 17 MIT. Scratch. <https://scratch.mit.edu/>. Accessed: 2021-03-08.
- 18 Sandra Mitre, Rodrigo Siqueira-Batista, José Girardi-de Mendonça, Neila Morais-Pinto, Cynthia Meirelles, Cláudia Pinto-Porto, Tânia Moreira, and Leandro Hoffmann. Metodologias ativas de ensino-aprendizagem na formação profissional em saúde: debates atuais. *Ciência & Saúde Coletiva*, 2(13):2133–2144, 2008.
- 19 José Morán. Mudando a educação com metodologias ativas. *Coleção Mídias Contemporâneas. Convergências Midiáticas, Educação e Cidadania: aproximações jovens*, 2:15–32, 2015.
- 20 João Silva, Gilvandenys Sales, and Juscileide Castro. Gamificação como estratégia de aprendizagem ativa no ensino de física. *Revista Brasileira de Ensino de Física*, 41(4), 2019.
- 21 Cinthia B. Spricigo. Estudo de caso como abordagem de ensino. *Revista Tuiuti: Ciência e Cultura*, 5(58), 2019.
- 22 Romero Tavares. Aprendizagem significativa, codificação dual e objetos de aprendizagem. *Revista Brasileira de Informática na Educação*, 18(2):4–16, 2010.

Moopec: A Tool for Creating Programming Problems

Rui C. Mendes   

Centro Algoritmi, Departamento de Informática, University of Minho, Braga, Portugal

Abstract

This paper presents a tool called Mooshak Problem Creator (Moopec) for facilitating the creation of programming exercises for a web-based multi-site programming contest system called Mooshak [7]. Users only need to create a text file for specifying all the information concerning problems including their description, tests and user feedback. This tool provides ways of automating most tasks involved in creating problems in Mooshak and, consequently, increases teachers' productivity. Moopec allows instructors to quickly create problem sets by simply editing a text file. Moopec is implemented in Python and is available at https://github.com/rcm/mooshak_problem_creator.

2012 ACM Subject Classification Applied computing → Computer-managed instruction

Keywords and phrases Automatic Program Assessment, Batch Generation, Testing

Digital Object Identifier 10.4230/OASICS.ICPEC.2021.9

Funding *Rui C. Mendes*: This research has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020.

Acknowledgements I want to thank Pedro Ribeiro for gently supplying the script used for providing the feedback shown in Fig. 1.

1 Introduction

Teaching students how to program is difficult [6]. There are several tasks of equal importance:

- (1) Supplying enough exercises for practice;
- (2) Providing feedback;
- (3) Summative assessment.

With the advent of the pandemic, there is an added difficulty: instructors and students do not share the same space and thus it is harder to help them in the classroom. Thus, the importance of using software tools that are capable of performing these tasks increased. Even though there are many problems available online, instructors still need to create new problems because most of the classical exercises have their solutions publicly available online. While this is extremely useful for learning how to program, it creates many difficulties for instructors when they are interested in summative assessment. Furthermore, when students submit their solutions to an online tool, instructors are able to address other concerns like code organization, cleanness, readability, efficiency or documentation.

When teaching programming, there are several kinds of exercises that can be given to students [1] involving all the categories of the Bloom taxonomy [13]. Most of these tasks detail several ways of evaluating students in written tests. However, in this work we are interested in programming exercises that may be given to students in practical classes and be evaluated by a programming assessment tool. The rationale behind this is to enable students to work at their own leisure, in class or at home. In this way, students may try the exercises, get automatic feedback and still be able to consult instructors either online or during class time. Programming exercises may fit into several categories [11, 9]:

code from scratch where students implement a solution to a problem given a description and a test suite that assesses their performance;



© Rui C. Mendes;

licensed under Creative Commons License CC-BY 4.0

Second International Computer Programming Education Conference (ICPEC 2021).

Editors: Pedro Rangel Henriques, Filipe Portela, Ricardo Queirós, and Alberto Simões; Article No. 9; pp. 9:1–9:7

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

9:2 Moopec: A Tool for Creating Programming Problems

implement an algorithm students are given an algorithm and are asked to implement it;

code skeleton students start with a skeleton that they have to fill, they may have to implement a few functions or classes;

code baseline the instructor supplies some code and the students have to extend it for a more general case or use it for solving a more complicated task;

jumbled code students are given all the lines of code but they are not in the correct order;

find the bug students are given buggy code and must find the bug, this is more useful when using quizzes and forces students to look at a solution, read it and try to understand why it doesn't work;

buggy code this is a generalization of the previous approach and allows students to improve their skills of code reading and understanding and can also help them become more proficient with a debugger, understand compilation errors, logic errors and problems handling edge cases ;

compiling errors this is also useful in quizzes as students are given a code snippet and the compiler error and must understand what the problem is

keyword use students are forced to use a given function or code and solve a task according to a strict specification (e.g., use `foldl` to find the maximum of a list of numbers).

There are many tools available for automatic assessment of programming exercises. Some are freely available online like codeboard [2] that provides a flexible way of creating problems and allows to implement many of the types of exercises given above. Furthermore, students can use an IDE for programming and can test their code locally or submit it. The author has used it in a course and the student's reception was good. In languages that have unit testing libraries, it provides ways of using unit tests to evaluate submissions. However, it is necessary to implement a fair portion of code for evaluating solutions in some programming languages (e.g., C). The instructor has to use a web based client and copy and paste code into the appropriate windows and create the necessary files one by one by using the WEB interface. While it is somewhat intuitive to use, its main drawback is that it does not have, as far as the author knows, a way of creating and importing problems in a batch fashion.

2 Mooshak for automatic assessment

Because of the effort necessary for creating large numbers of problems efficiently in the tools presented above, the author prefers using Mooshak [7] for automatic assessment. This is mainly due to the fact that instructors install it on a machine they control and can use the file system for creating contests, copying them, copying problems between contests and setting them up by editing text files. Mooshak has its own internal format to describe problems called Mooshak Exchange Format (MEF). MEF includes a XML manifest file referring several types of resources such as problem statements (e.g. PDF, HTML), image files, input/output test files, correctors (static and dynamic) and solution programs. The manifest also allows the inclusion of feedback and points associated to each test [10, 8]. Mooshak has been used on ICPC and IOI contests for many years as well as for education in several universities. The administrator can create several kinds of contests like:

IOI the format used in the International Olympiads in Informatics [5], where students are given points for each test they pass

ICPC the format used in the International Collegiate Programming Contest [4]

short the format used in code golf

Since its inception, Mooshak has evolved and has provided some improvements that facilitate its use in educational activities. It is possible to specify how much CPU time is necessary to solve the problem, how many resources are used and assign a point value for

each test or show a feedback message if the user fails a given test or even to show the test to the user. Each test has:

- an input, which is a text file that is fed to the program's *standard input*;
- an output, which is a text file that is compared using `diff` to the program's *standard output*;
- arguments, that are passed to the program;
- a point value, that is awarded if the submission gives the intended output for this test;
- a feedback message and
- a text file containing a context that can be used by Mooshak's dynamic evaluator for the given problem.

On the surface, it seems that Mooshak may only be used for exercise types like *code from scratch* but it may actually be used for more categories with a little imagination. Mooshak allows the administrator to configure more programming languages and to supply static or dynamic correctors for problems.

When creating a programming language, the administrator provides among others:

- (1) What is the extension of the file submitted by the user
- (2) How to *compile* the program
- (3) How to run the program

When compiling the program, the administrator can supply the compiler with information about the team, the program environment, the extension and a solution to the problem. The static corrector receives the source code, a solution and a possible program environment and can classify the submission by simply looking at the source code. The dynamic corrector is invoked by Mooshak after it has run the program and receives information about the test like the input, expected and obtained output, the standard error contents, Mooshak's classification and the context given.

Thus, it is possible to implement some of the categories supplied above:

code from scratch this one is self-evident

implement an algorithm this category may be possible if the algorithm is the only one capable of solving the problem within a given order of complexity by careful creation of the tests and specification of the timeout in CPU seconds or the memory bound needed to solve each test

code skeleton the *compiler* may incorporate the students' solution in a larger program and thus guarantee that they actually used the skeleton given; alternatively, a static corrector could be used to check for the existence of the skeleton

code baseline same as the previous category

jumbled code the static corrector may be used for ensuring that the students didn't modify the program but simply supplied the lines in the correct order

find the bug the static corrector may be used for checking if the students are submitting the code with the bugs corrected

buggy code this is similar to the previous category

compiling errors this one may also be handled like the previous alternatives

keyword use a static corrector may help, or a specific language that logs the use of the function or keyword in the manner expected

3 Creating problems in Mooshak using the WEB interface

The difficulty involved in creating a new programming problem involves:

- (1) providing the problem's name, letter and description
- (2) deciding specific limits (e.g., CPU, memory)
- (3) supplying the tests' input and output
- (4) decide how much points each test is worth
- (5) indicate whether it should be shown to the user

The first and second steps usually involve using a system that either generates HTML or PDF and then uploading the file containing the description to Mooshak through the web interface along with filling the other required fields. The remaining tasks are performed for each test using the web interface. In the current year, the author has used Mooshak in several courses, one of them with around 250 students, and has created around fifty new problems with varying degrees of complexity, some with 100 different test cases. In many of these cases, tests address different problems and thus have to be organized in several categories, with different feedback messages and points. When talking about these numbers, it is very important to have a tool that is able to help create tests in an automatic or semi-automatic manner.

4 Creating a new problem using the tool

The solution proposed here is to use a domain specific language (DSL) for creating problems by using a text editor. In order to create a problem or set of problems, the user needs to create a text file using a simple textual format. The user may specify the following fields:

- NAME** This is the first keyword given and supplies the name of the problem; the problem ends with the corresponding **END** keyword
- LETTER** This is the name of the folder that stores the problem and what is stored in letter
- TESTS** One or more tests; it ends with an **END** keyword and may have the following fields:
- INPUT** One line of input
 - LONGINPUT** One or more lines of input; it ends with an **END** keyword
 - INPUTGEN** An optional integer n followed by a function (either a lambda function or the name of a function in an imported module) that will be used to generate n inputs
 - FEEDBACK** A feedback message for these tests
 - POINTS** How many points to award to each of these tests
 - SHOW** If given, these tests will be shown
- IMPORT** The name of a module to import
- CODE** One or more lines, ending with an **END** keyword. This creates a module and adds the code into it
- SOLVER** This may be either a lambda function or the name of a function in an imported module and will be used to create the outputs
- DESCRIPTION** One or more lines, using the markdown syntax, ended with the **END** keyword
- POINTS** If given it evenly distributes this number of points to all tests

Listing 1 shows a contrived example of a problem. The first two tests have feedback and are shown to the user. The following test is completely blind: it neither has any feedback nor is shown to the user. The following 10 tests are generated automatically and have a feedback and are shown. The last five tests are somewhat larger and also have a feedback but are not

shown. The function that solves this problem was supplied by a Python lambda function. The description uses the markdown format supplied by the markdown module [12]. All these tests sum to 100 points that are evenly distributed among them. In this case, all the code is contained in the file instead of importing a module. It is possible to import several modules, one per IMPORT keyword and also supply several CODE keywords.

■ **Listing 1** An example.

```

NAME      Summing a bunch of  numbers
LETTER    A
CODE
def gen_list(size, min_val, max_val):
    import random
    return lambda: ' '.join(str(random.randint(min_val, max_val))
                             for i in range(size))
pequeno = gen_list(10, 1, 10)
medio = gen_list(100, 1, 100)
END
SOLVER   lambda s: sum(int(x) ** 2 for x in s.split())
TESTS
        INPUT    10 20 30
        LONGINPUT
12
13
END
        FEEDBACK      Examples given in the problem description
        SHOW
END
TESTS
        INPUT    15 25 35
END
TESTS
        INPUTGEN 10 pequeno
        FEEDBACK small tests
        SHOW
END
TESTS
        INPUTGEN 5 medio
        FEEDBACK larger tests
END
DESCRIPTION
# Sum a list of numbers
Create a program that:
- reads several numbers and
- prints the sum of their squares
END
POINTS 100
END

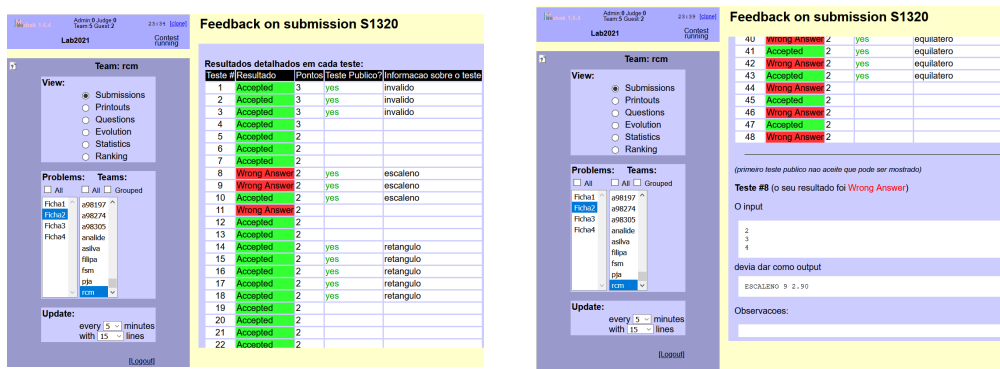
```

A file may have more than one problem. Each problem starts with the NAME keyword and ends with the corresponding END keyword. In order to use the system, one simply has to execute the command `moopec` with the files as arguments:

```
moopec example.txt
```

which will create one folder with the corresponding letter for each problem. Then it is just a matter of copying these folders into the Mooshak's problem directory inside the corresponding contest and adjust the permissions accordingly.

9:6 Moopec: A Tool for Creating Programming Problems



■ **Figure 1** Example of feedback of a problem in Mooshak.

Figure 1 shows an example of the feedback of a problem generated using this system inside Mooshak ¹. This problem has 48 tests and had 1775 submissions. The goal is to find the type, perimeter and area of a triangle given the lengths of its sides.

5 Conclusions

Moopec aims to facilitate the batch creation of problems for the Mooshak system. Since this system has been used to host several ICPC and IOI events for many years, it is robust and can be used extensively for hosting programming assignments. Moopec facilitates the batch creation of programming exercises. It not only allows instructors to handcraft tests but also to use generators that are able to create tests for evaluating specific concerns. Given that Mooshak can be used for many categories of programming assignments and Moopec helps automate the creation of these exercises, it helps instructors be more productive and easily create new assignments, adapt older ones or combine them.

It may seem that one of the vulnerabilities of the system is that the solution can only be implemented in Python. However, since Python can easily call other programs using `os.system` or `subprocess`, this is not a significant drawback. The main advantage of this tool is to simplify the task of creating a problem set into that of creating a text file. While this may not seem to be as intuitive as some of the usability design principles advocate, it greatly speeds up the task of creating problem sets and thus greatly increases the instructor's productivity.

This tool has already been used in practice for creating several kinds of programming assignments, ranging from simple programming exercises to evaluating full-fledged projects. In the current year, the author of this work has used Moopec for creating 9 contests with close to 50 different problems. Many of these problems had around 50 tests, with some having 100 tests and most of them had a message documenting what was the situation addressed by the tests. While some of the tests were handcrafted for testing specific purposes, others used generators for checking specific concerns. The contests in the second semester of 2021 are being used by students with over 8000 submissions and will probably be closer to 9000 until the end of the semester. One of these contests is a project where the users submit a zip file containing C code that is then compiled and run.

¹ The feedback code was gently provided by Pedro Ribeiro.

Even though the feedback could be improved, the current version is already quite useful. It allows partial evaluation, i.e., it uses an IOI grading philosophy and provides feedback that helps users understand which tests failed. In the project assignment mentioned above, all the tests were public and thus the students are able to understand which conditions caused their program to fail and thus try to address the problem. The sheer amount of tests deters students from trying to cheat the system by simply creating code that passes all the tests. While this philosophy has its problems as have other similar ones [3], it is an invaluable tool when trying to create tasks for continuous evaluation of students in an online context without overburdening instructors with the gigantic task of evaluating all these submissions and providing adequate feedback. Future work will add extensions to the DSL in order to provide static and dynamic correctors.

References

- 1 Matt Bower. A taxonomy of task types in computing. In *Proceedings of the 13th annual conference on Innovation and technology in computer science education*, pages 281–285, 2008.
- 2 Codeboard. <https://codeboard.io/>. Accessed: 2021-04-07.
- 3 Michal Forišek. On the suitability of programming tasks for automated evaluation. *Informatics in education*, 5(1):63–76, 2006.
- 4 International collegiate programming contest. <https://icpc.global/>. Accessed: 2021-04-07.
- 5 International olympiads in informatics. <https://ioinformatics.org/>. Accessed: 2021-04-07.
- 6 Tony Jenkins. On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, volume 4, pages 53–58. Citeseer, 2002.
- 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 Carlos Paiva, Ricardo Queirós, José Paulo Leal, and Jakub Swacha. Yet Another Programming Exercises Interoperability Language (Short Paper). In Alberto Simões, Pedro Rangel Henriques, and Ricardo Queirós, editors, *9th Symposium on Languages, Applications and Technologies (SLATE 2020)*, volume 83 of *OpenAccess Series in Informatics (OASICs)*, pages 14:1–14:8, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/OASICs.SLATE.2020.14.
- 9 Dale Parsons and Patricia Haden. Parson’s programming puzzles: a fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*, pages 157–163, 2006.
- 10 Ricardo Queirós and José Paulo Leal. Babelo—an extensible converter of programming exercises formats. *IEEE Transactions on Learning Technologies*, 6(1):38–45, 2012.
- 11 Alberto Simões and Ricardo Queirós. On the Nature of Programming Exercises. In Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões, editors, *First International Computer Programming Education Conference (ICPEC 2020)*, volume 81 of *OpenAccess Series in Informatics (OASICs)*, pages 24:1–24:9, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/OASICs.ICPEC.2020.24.
- 12 Manfred Stienstra, Yuri takhteyev, Waylan limberg, and Waylan Limberg. Python-markdown. <https://pypi.org/project/Markdown/>. Accessed: 2021-04-07.
- 13 Errol Thompson, Andrew Luxton-Reilly, Jacqueline L Whalley, Minjie Hu, and Phil Robbins. Bloom’s taxonomy for cs assessment. In *Proceedings of the tenth conference on Australasian computing education-Volume 78*, pages 155–161, 2008.

Automated Java Challenges' Security Assessment for Training in Industry – Preliminary Results

Luís Afonso Casqueiro ✉ 

University Institute of Lisbon, (ISCTE-IUL), ISTAR, Portugal

Tiago Espinha Gasiba ✉ 

Siemens AG, Munich, Germany

Maria Pinto-Albuquerque ✉ 

University Institute of Lisbon (ISCTE-IUL), ISTAR, Portugal

Ulrike Lechner ✉ 

Universität der Bundeswehr München, Munich, Germany

Abstract

Secure software development is a crucial topic that companies need to address to develop high-quality software. However, it has been shown that software developers lack secure coding awareness. In this work, we use a serious game approach that presents players with Java challenges to raise Java programmers' secure coding awareness. Towards this, we adapted an existing platform, embedded in a serious game, to assess Java secure coding exercises and performed an empirical study. Our preliminary results provide a positive indication of our solution's viability as a means of secure software development training. Our contribution can be used by practitioners and researchers alike through an overview on the implementation of automatic security assessment of Java CyberSecurity Challenges and their evaluation in an industrial context.

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

Keywords and phrases Education, Teaching, Training, Awareness, Secure Coding, Industry, Programming, Cybersecurity, Capture-the-Flag, Intelligent Coach

Digital Object Identifier 10.4230/OASICS.ICPEEC.2021.10

Funding *Maria Pinto-Albuquerque:* This work is partially financed by national funds through FCT - Fundação para a Ciência e Tecnologia, I.P., under the projects FCT UIDB/04466/2020 and UIDP/04466/2020. Furthermore, the first and third author thank the Instituto Universitário de Lisboa and ISTAR, for their support.

Acknowledgements The authors would like to thank all the survey participants for taking part in this preliminary study, and for their helpful and constructive feedback. Furthermore, the authors would like to thank the hosting organization for enabling the study to take place.

1 Introduction

Over the last years, the number of cybersecurity incidents has been continually rising. According to the US department of homeland security [7], the root cause of about 90% of these incidents is due to poor software quality, which results in software vulnerabilities. A recent large-scale study by Patel et al. [17] has shown that more than 50% of software developers cannot identify vulnerabilities in source code. These facts pose serious issues in the industry, especially for companies that deliver products to critical infrastructures. Companies can use several methods to improve software quality, such as code reviews and static application security testing (SAST). Another possibility is to address the human factor, i.e., the software developers, through education on secure coding.



© Luís Afonso Casqueiro, Tiago Espinha Gasiba, Maria Pinto-Albuquerque, and Ulrike Lechner; licensed under Creative Commons License CC-BY 4.0

Second International Computer Programming Education Conference (ICPEEC 2021).

Editors: Pedro Rangel Henriques, Filipe Portela, Ricardo Queirós, and Alberto Simões; Article No. 10;

pp. 10:1–10:11



OpenAccess Series in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

10:2 Automated Security Assessment of Java CyberSecurity Challenges

In this work, we explore raising awareness on secure coding using serious games. Previous work by Gasiba et al. [10] identifies serious games as a viable alternative to raise secure coding awareness of software developers in the industry. However, their work does not address Java programmers. Java is not only widely used in the industry and taught at universities, but it is also in the top-3 of the programming languages that have the most open source vulnerabilities [18].

In this work, we propose and implement a method to perform an automatic evaluation of the security level of Java challenges. This process generates hints for the players when the code contains vulnerabilities. Our platform is embedded in a serious game called CyberSecurity Challenges [11] that has the goal of raising awareness of software developers in the industry. This game is inspired in the capture-the-flag genre of games and includes several types of challenges, both offensive and defensive in nature. Once a team solves one of the challenges they receive a flag earning them a certain amount of points. In the end the team with most points wins the Cyber Security Challenges event. No one wins if no one passes all the tests. However, the main goal of the game is to raise awareness of secure coding, and winning the game by collecting points serves only as an incentive for the individual players and teams to actively participate in the game.

The designed java challenge's artifact, in the present work, allows for a fully automatic interaction between players through the hint system. The hints are designed to raise awareness of the Carnegie Mellon's Java secure coding guidelines [22]. Our research questions are:

RQ1 How to automatically assess the level of security of a player's Java code?

RQ2 Which open-source tools can be used to assist the Java security code assessment?

RQ3 How do players perceive the platform as a means to learn Java secure coding?

The main contribution of this work is insight for practitioners and researchers who wish to implement Java secure coding challenges with automatic hint generation. Moreover, this work also provides a relevant method to automatically access the security levels of Java code, a list of relevant tools and an analysis in the industry setting. Additionally, the empirical study and preliminary results provide a further contribution to knowledge, and its raw results are provided in Zenodo [14] to enable further research.

The present work is organized as follows; section 2 discusses previous relevant work, section 3 describes how to perform automatic security assessment of Java challenges, and the tools we use. In section 4, results of a preliminary study in the industry is presented, and section 5 concludes this work.

2 Related Work

In [15], Meng et al. identified several security-related problems that Java software developers experience when developing software. In particular, they conclude that developers do not understand the security implications of their coding decisions. Also, developers tend to search online forums for answers. However, Fischer et al. [9] have shown that such information can be outdated and possibly even wrong. Oliveira et al. [16] discuss the lack of awareness by software developers of the specification and proper usage of application programming interfaces (API). They conclude that wrong usage of APIs can lead to security vulnerabilities. Gasiba et al. identify that software developers in the industry lack awareness [13] of secure coding guidelines, e.g., [22]. The result of their study is in agreement with a recent large-scale survey, with more than 4000 software developers, by Patel et al. [17] that concludes that more than 50% of software developers cannot identify vulnerabilities in source code.

One possible way to raise awareness of secure coding is through serious games. Dörner et al. [8] define serious games as a game that is *designed with a primary goal and purpose other than pure entertainment*. Culliane et al. [6] argue that these types of games can be educational and fun, and Sorace et al. [24] conclude that these can be an attractive approach to improve awareness. Graziotin et al. [12] show that *happy developers are better coders*. Furthermore, Gasiba et al. developed C/C++ CyberSecurity Challenges for industrial software developers [11]. In this work, we adapted this platform to provide Java CyberSecurity Challenges.

Several serious games have been developed with success. Bakan et al. [1] provide an overview of current research in game-based learning and teaching environments. They conclude that serious games can be an effective teaching tool in terms of students' achievements and retention. In a similar work, Cardoso et al. [2] propose to integrate a Virtual Programming Lab in Moodle to provide a platform for students to edit, compile, run, debug and evaluate programs. However, their work is performed in academia, focuses on teaching Java programming to undergraduates, and does not focus on secure coding.

To design a serious game for secure programming in Java, which produces automatic hints, a method to automatically evaluate a player's code in terms of cybersecurity vulnerabilities needs to be developed. However, to the best of our knowledge, previous literature does not address these combined issues.

The present work uses and is embedded in a study that uses Action-Design Science methodology by Sein et al. [21]. This research methodology addresses real-world problems in organizations through meaningful actions which are informed by cooperation with academia.

3 Automated Security Assessment of Java Challenges

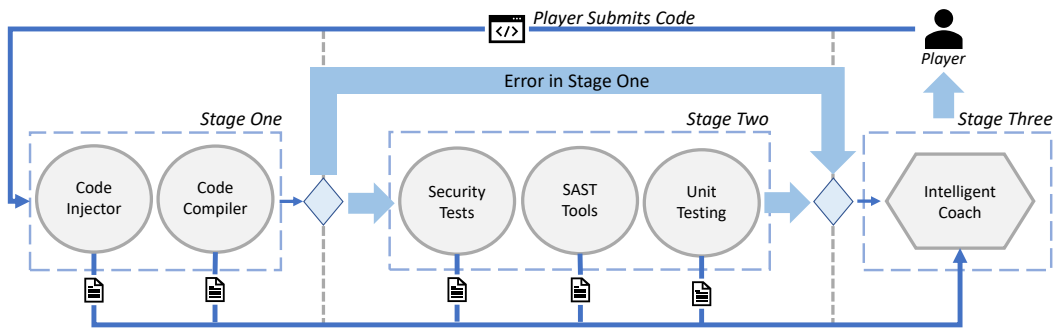
This section briefly describes the automated security assessment process, how the implementation-related security problems are addressed, and describes an empirical study performed in the industry.

3.1 Automated Security Assessment Process

A Java challenge consists of the following phases. First, the player is presented with a challenge (programming exercise) through a web interface containing at least one cybersecurity vulnerability. The player's task is to identify the vulnerability in the code and rewrite it such that the rewritten code still implements the challenge's required functionality. When the player is satisfied with the rewritten code, he or she submits it to the backend. At this point, the backend performs several analysis steps to the submitted code to automatically assess it in terms of vulnerabilities and functionality. If the code contains no vulnerabilities and is functionally correct, the player has won the challenge. Otherwise, the backend generates a hint to send back to the player to help them solve the challenge. The hint's generation is based on the player's code and the various analysis steps performed during code analysis. Furthermore, the hints are implemented using a laddering technique [19, 11].

Figure 1 details the various stages undertaken during the submitted code's automated assessment. These stages are composed of several steps. The first stage includes two steps: code injection and compilation. When needed, the players' code is modified through code injection. This step might be required to insert additional functionalities, such as new classes into the project, through import statements without the player's knowledge. Next, the resulting code is compiled. If any errors are encountered, the process jumps to stage three. In the absence of compiler errors, the process advances to stage two.

10:4 Automated Security Assessment of Java CyberSecurity Challenges



■ **Figure 1** Automatic Assessment Process.

There are two major possible approaches to stage two. The tools employed can either run in parallel or sequentially. In this case we used a sequential process due to the fact that the amount of tools used plus the correspondent execution time does not justify a parallel approach. This means that the difference in the response time does not substantially improve from one approach to the other.

This stage consists of code tests using three different testing methods: security tests, static code analysis, and unit tests.

Unit testing is performed to ensure that the player’s code works according to the challenge’s specifications. Security tests are dynamic unit tests that mimic malicious user actions and input to exploit the exercise’s vulnerability. Static-application security testing (SAST) tools provide a report based on analysis of the player’s code that includes code vulnerability issues. These tools however can produce wrong evaluations of the given code.

■ **Table 1** SAST Tools Outcomes.

Type	Description
True Positive	Code has a vulnerability and the tool flags it
True Negative	Code doesn’t have a vulnerability, and the tool does not flag it
False Positive	Code doesn’t have a vulnerability, but the tool flags it
False Negative	Code has a vulnerability, and the tool doesn’t flag it

Table 1 shows the possible outcomes of using SAST tools. Of the four possibilities only two represent the desired functionality of these tools, True Positive and True Negative. In our work, we only consider a sub-set of the results reported by the tools, in order to minimize the number of false positives and false negatives in our security assessment.

Table 2 summarizes the tools that we have used in this stage and their mapping to the testing methods. These tools were selected based on the capability to answer the stage two needs of testing. For the SAST tools we gathered industry recognized free and open-source software and then filtered by the amount of security related errors. The JUnit tool, a framework widely used in industry, was implemented for unit-testing of the submitted code. This tool ensures that the player’s code works according to the specifications and goals of the challenge. The unit-tests will flag issues like empty class (where no security vulnerabilities are found), giving the proper feedback to the player. JUnit is also used in security-testing of the code. This tool provides extreme values in the functions call in order to try and trigger any potential issue present in the program. JavaFuzzer and Spoon are other tools utilized

■ **Table 2** Analysis Tools.

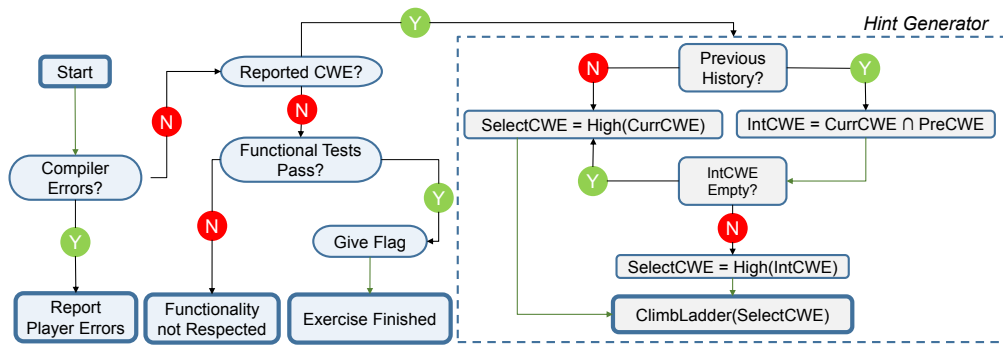
Tools	Unit Tests	Security Tests	SAST Tools
JUnit	•	•	
JavaFuzzer		•	
Spoon		•	
Self-Developed Tools		•	
SonarQube			•
SemGrep			•
FB-Infer			•
SpotBugs			•
PMD			•
JBMC			•

for security-testing. These tools provide, among others, a series of tests that stimulate the given code by creating random inputs in order to try and trigger potential errors in the given program [20].

The employment of several different testing tools and methods offers a greater variety of results. Although many times different testing tools flag the same errors, there are also several cases where only specific tools can spot certain security vulnerabilities. This means that the tools used can sometimes overlap but also complement each others in the results of the analysis, leading to a more complete scrutiny of the player's code. These tools are used to generate security-related findings of the submitted code. The tools' findings are mapped to a PASSFAIL boolean value and to a specific common weakness enumeration (CWE) [3]. The mapping to individual CWEs as well as its ranking was done according to our experience in cybersecurity. Note that, since the user is free to change the exercise's code at will, potentially malicious code will be executed in stage two. Therefore, to prevent malicious players from abusing and breaching our system (intentionally or non-intentionally), the security tests and unit tests run in a time-limited sandbox. The time-limitation protects against infinite loops, and the sandbox protects against remote-code execution. For more information, we refer the reader to [11].

The last stage consists of an artificial intelligence engine that creates relevant hints for the player if the challenge is not solved. If the challenge is solved, the last stage awards points to the player. Hints are generated based on a CWE selected from the previous steps' results and prior history, using a laddering technique. The selection of the CWE, which is supplied to the laddering technique, is shown in figure 2.

In case of compiler errors, no CWE is selected, and a report is sent back to the player, which includes the compiler error message. If the code compiles with no errors but no CWE is reported from the tools, the algorithm checks the functional tests' results. If all the tests pass, the player wins the game, is given a flag in recognition, and the exercise is finished. If at least one test fails, the player receives feedback based on the failed functional test case. This feedback states that the desired code functionality is not respected. However, when the code has security issues (i.e., at least one CWE is reported), the hint generator starts. The hint-generation checks if there have been previous hints given to the player based on the prior history. If the prior history is empty, the hint will be generated based on the highest-ranking CWE found out of all the reported CWEs. This CWE ranking was pre-determined by us based on our experience.



■ **Figure 2** Intelligent Coach’s Algorithm.

■ **Table 3** Initial Hints.

Hint Type	Description
Overall Feedback	Resource Leakage Found. The submitted code has some type of resource leakage present.
Basic Hint	CWE-772 : The software does not release a resource after its effective lifetime has ended, i.e., after the resource is no longer needed.
First Hint	You must guarantee that the stream is being closed in every possible scenario.

However, if the prior history is not empty, the intersection between the current set of CWE’s and the ones in the prior history is computed. If the intersection is an empty set, the selected CWE will be computed based on the highest-ranked CWE out of all the reported CWEs. If the intersection is not empty, the selected CWE is computed based on the highest-ranked CWE in the intersection list.

Finally, the hint is then generated based on the selected CWE through a laddering technique. More details on the laddering technique are given in [11, 19].

Table 3 shows the initial hints developed for the initial platform. Each time the player submits the code the platform will release the Overall Feedback and a Hint. This feedback message is a generalized view of the issue found. The Hint given starts with the associated description of the reported CWE and evolves to more specific hints over time. In our instance of the security exercise, the basic hint provides a description of the vulnerability CWE-772 [5]. In the next hint level, a directed message on how to solve the exercise is provided to the player.

In our project, we took the decision to consider security issues with higher priority than functional issues. This means that the generated hints are firstly concerned with security issues and, if no security issues are present, they are concerned with code functionality.

The whole process of analysing submitted code and generating the hints, executed each time a player submits code, usually takes no more than one or two seconds. These time measurements were acquired using a laptop with the following specifications: 12GB of RAM with an Intel Core i5-6200U CPU, running Linux xubuntu version 20.04.

3.2 Empirical Study

We conducted an empirical study using the developed prototype to analyze how the players feel about the platform as a cybersecurity learning system. The test was conducted from 22 to 27 February 2021 through individual online meetings, 11 in total. Each meeting lasted 30 minutes, 20 of those for the solving of the challenge and 10 for a brief interview with the participants. Eleven persons participated in our experiment, with ages ranging from 20 to 35. Nine of the participants are working in the industry, and the remaining two were students in the IT/software development fields. Industry participants are from Germany and Portugal and are software developers for critical infrastructures with more than five years of experience. The academia students were last-year undergraduates from Portugal, studying computer science.

The players received one Java challenge with a security vulnerability to solve. The individual meetings were carried out with each individual participant. While not embedded in a CyberSecurity Challenges event, the present work is used to inform our approach on the implementation of secure coding exercises for the Java programming language.

■ **Listing 1** ResourceLeak.java.

```
import java.io.*;

public class ResourceLeak {
    public void writeToFile(File file, String msg) throws IOException {
        FileOutputStream fos = new FileOutputStream(file);
        fos.write(msg.getBytes());
    }
}
```

Listing 1 shows the source code of the exercise presented to the participants during our study. This code was based on the FIO04-J rule [23] of the SEI-CERT coding standards, and contains a resource leakage vulnerability. This type of security issue arises when an opened file handle is not closed. Although Java is a garbage-collected language, the garbage collection mechanism does not work for file handles and database connections. Once these resources are no longer needed they should be immediately closed. A failure to do so can lead to resource starvation of the program or in critical cases resource exhaustion, ultimately leading to a denial-of-service attack [4]. To solve this exercise the participants need to close the file handle either by using a *try-catch-finally* Java block or by implementing a *try with Resources*. In the *try with Resources* scenario the FB Infer tool (V1.0.0) suffers from a False Positive that flags a resource leakage in the code. This problem can be initially addressed in two ways. The first is to wait for the tool to receive an update that would resolve this issue. The second, and chosen course of action, was to give the flag of the challenge on the off chance that the players would encounter this scenario. The participants tested and solved the Java exercise, and experimented with the platform without any restrictions. After solving the challenge, they were asked to complete a small survey containing questions related to their experience, and additional open discussions were held. The survey questions are detailed in table 4.

Answers to the questions are based on a 5-point Likert scale from 1-strongly disagree to 5-strongly agree. Answers were collected anonymously, the participants were instructed about the goal of the research being carried, and they agreed to take part in the study.

■ **Table 4** Survey Questionnaire.

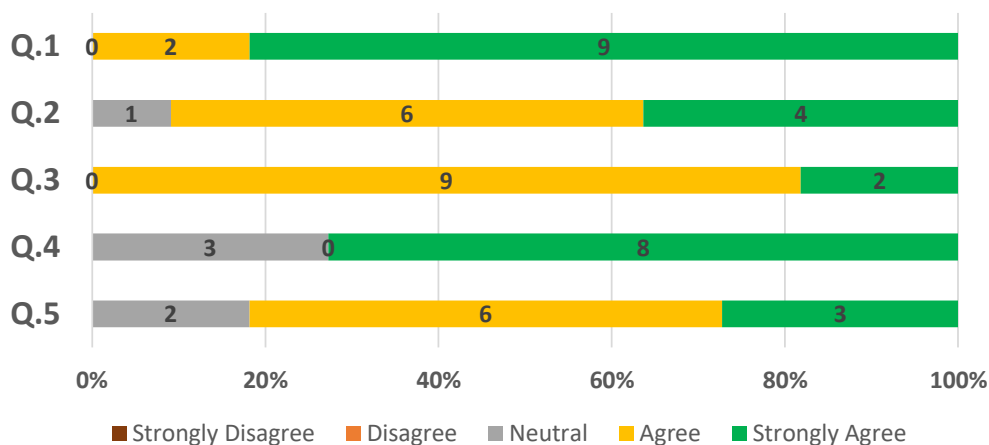
Identifier	Question
Q.1	The error messages and hints issued by the platform are relevant to the exercise.
Q.2	I can relate the hints to the code I have written.
Q.3	The hints provided by the platform make sense to me.
Q.4	If I am given the opportunity, I would like to play more exercises.
Q.5	The hints helped me to understand the problem with the code I have written.

4 Evaluation

In this section, we present the results of our empirical study. Moreover, a critical discussion of the results is also presented. Additionally, we discuss possible threats to the validity of our conclusions. The raw results of our survey can be found in Zenodo [14].

4.1 Results and Discussions

The present work answers the first research question through the presented methodology of using the CyberSecurity Challenges platform to implement Java exercises. In particular, figure 1 shows the three-stage method that we have implemented to automatically assess the level of security of the Java code from the players to the game. To answer the second research question, we have evaluated a number of openly available tools. These results are summarized in table 2. In the following, we present a discussion related to the third research question, on how the players perceive the platform as a means to learn secure coding in the Java programming language.



■ **Figure 3** Results of Empirical Study in the Industry.

Figure 3 shows the results of our empirical study, in relation to the survey questions (see table 4). Our preliminary results show that all participants either agreed or strongly agreed that the platform’s hints make sense to them (Q.3) and are related to the programming exercise (Q.1). The majority of the participants agree or strongly agree that the hints are related to the code they wrote (Q.2), with only one participant giving a neutral answer. More than 80% of the participants agree or strongly agree that the platform’s hints help to understand the security issues present in the code (Q.5). About 72% of the participants strongly agree that, given the opportunity, they would like to play additional exercises (Q.4); however, three participants (27%) are not sure.

Although Q.2, Q.4, and Q.5 show mostly positive results, one, three, and two neutral answers were obtained for each question, respectively. Further studies are needed to understand possible reasons that led the participants to give these answers. Overall, the outcome obtained in this preliminary study, both through the survey and additional discussions, indicates that the participants welcome the exercises. Furthermore, the hints generated by our proposed automatic security assessment method, which uses widely available open-source tools, received positive ratings from the participants of our study. This is in line with previous similar studies for other programming languages, and gives the authors encouragement to further develop the platform.

Although most participants come from the industry, we hypothesize that similar results might be obtained for students in academia. Furthermore, we also hypothesize that the same method can raise awareness on other coding issues, e.g., code-smells, naming conventions, and code format.

This paper is focused on the Java programming language and the security vulnerabilities associated with it. However, the method to automatically assess the submitted code can be applicable and extendable to any programming language. Just, for each programming language the set of vulnerabilities is different and thus the assessment method needs to be updated accordingly. Although these tests are being developed for the industry, they can also be applied in academia. The developed artifact can be incorporated in programming courses (mainly junior-level) to teach these concepts to students, possibly leading to better prepared and security-aware software developers. The purpose of the artifact itself can be shifted. Instead of focusing 100% on the security part of code development, it can be enhanced to deal with and cover other issues, e.g., code-smells or naming convention.

4.2 Threats to Validity

Eleven participants took part in our preliminary empirical study, which is in line with similar preliminary empirical studies in industrial settings. The results were collected anonymously, and the gathered data does not allow us to understand the difference between participants from industry and students. Nevertheless, since the overall result is positive, we believe that an eventual bias does not affect our conclusions. Furthermore, our results are in alignment with previous studies. Static-application security tools are known to produce false-positives and false-negatives. To counteract these issues, we use the following strategy: (1) our platform filters the findings based on our tools' experience, (2) we use additional security-tests to cover false-negatives, and (3) we test the exercise extensively before deployment.

5 Conclusions and Further Work

Over the past decade, the number of cybersecurity incidents has been increasing. To counteract this issue, companies can follow several strategies to reduce the number of vulnerabilities in their products and services. These strategies aim to increase resilience to malicious attacks. In this work, we focus on the human-factor, through awareness training. To achieve this, we adapt and extend previous work on CyberSecurity Challenges to automatically perform security assessment of Java challenges and generate hints for players. This paper gives a brief overview of how this automatic assessment can be performed using openly available tools enabling practitioners to reproduce our results. We also introduce a hint-generation algorithm that is used to interact with the player. Finally, we perform an empirical evaluation of our proposed method together with eleven participants. Nine participants are working in the industry, and two are students in the software development

field. Our preliminary results show that the participants can understand the hints generated through our method and agree that these hints are helpful to solve the Java challenge. In further work, the authors would like to implement additional Java challenges and perform a detailed empirical evaluation of the artifact in an industry setting. In this future work, we would like to understand how the hint system targeting Java challenges can be improved, in particular addressing to which extent and which strategies can cope with the fact that findings from static-application security testing tools can potentially contain false-positives and false-negatives.

References

- 1 Uğur Bakan and Ufuk Bakan. Game-Based Learning Studies in Education Journals: A Systematic Review of Recent Trends. *Actualidades Pedagógicas*, pages 119–145, July 2018. doi:10.19052/ap.5245.
- 2 Marílio Cardoso, António Vieira de Castro, Álvaro Rocha, Emanuel Silva, and Jorge Mendonça. Use of Automatic Code Assessment Tools in the Programming Teaching Process. In Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões, editors, *First International Computer Programming Education Conference (ICPEC 2020)*, volume 81 of *OpenAccess Series in Informatics (OASICs)*, pages 4:1–4:10, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/OASICs.ICPEC.2020.4.
- 3 MITRE Corporation. Common Weakness Enumeration. Online, Accessed 4 July 2019. URL: <https://cwe.mitre.org/>.
- 4 MITRE Corporation. Common Weakness Enumeration - 404. Online, Accessed 4 July 2019. URL: <https://cwe.mitre.org/data/definitions/404.html>.
- 5 MITRE Corporation. Common Weakness Enumeration - 772. Online, Accessed 4 July 2019. URL: <https://cwe.mitre.org/data/definitions/772.html>.
- 6 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.
- 7 Department of Homeland Security, US-CERT. Software Assurance. Online, Accessed 27 September 2020. URL: <https://tinyurl.com/y6pr9v42>.
- 8 Ralph 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.
- 9 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.
- 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 Tiago Gasiba, Ulrike Lechner, and Maria Pinto-Albuquerque. Sifu - a cybersecurity awareness platform with challenge assessment and intelligent coach. In *Special Issue of Cyber-Physical System Security of the Cybersecurity Journal*, Online, October 2020. SpringerOpen.
- 12 Daniel Graziotin, Fabian Fagerholm, Xiaofeng Wang, and Pekka Abrahamsson. What happens when software developers are (un)happy. *Journal of Systems and Software*, 140:32–47, June 2018. doi:10.1016/j.jss.2018.02.041.
- 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 Luis Afonso Casqueiro. Automated Java Challenges' Security Assessment for Training in Industry – Preliminary Results. *Zenodo*, February 2021. . doi:10.5281/zenodo.4740829.
- 15 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.
- 16 Daniela Seabra Oliveira, Tian Lin, Muhammad Sajidur Rahman, Rad Akefirad, Donovan Ellis, Eliany Perez, Rahul Bobhate, Lois A DeLong, Justin Cappos, and Yuriy Brun. API Blindspots: Why Experienced Developers Write Vulnerable Code. *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, pages 315–328, August 2018. (USENIX) Association, Baltimore, MD, USA, ISBN: 978-1-939133-10-6. URL: <https://www.usenix.org/conference/soups2018/presentation/oliveira>.
- 17 Suri Patel. 2019 Global Developer Report: DevSecOps Finds Security Roadblocks Divide Teams. Online, Accessed 18 July 2020. URL: <https://tinyurl.com/3z57t32d>.
- 18 Alison DeNisco Rayome. The 3 Least Secure Programming Languages, March 2019. URL: <https://www.techrepublic.com/article/the-3-least-secure-programming-languages/>.
- 19 Tim Rietz and Alexander Maedche. LadderBot: A Requirements Self-Elicitation System. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pages 357–362. IEEE, 2019.
- 20 Marc Schönefeld. *Java-Security: Sicherheitslücken identifizieren und vermeiden*. MITP-Verlags GmbH & Co. KG, 2011.
- 21 Maung Sein, Ola Henfridsson, Sandeep Puro, Matti Rossi, and Rikard Lindgren. Action Design Research. *MIS Quarterly*, 35(1):37–56, March 2011. doi:10.2307/23043488.
- 22 Software Engineering Institute, Carnegie Mellon. SEI CERT Oracle Coding Standard for Java. Online, Accessed 11 June 2018. URL: <https://tinyurl.com/ypm4mnj8>.
- 23 Software Engineering Institute, Carnegie Mellon. SEI CERT Oracle Coding Standard for Java - FIO04-J. Release resources when they are no longer needed. Online, Accessed 11 June 2018. URL: <https://wiki.sei.cmu.edu/confluence/display/java/FIO04-J.+Release+resources+when+they+are+no+longer+needed>.
- 24 Silvio Sorace, Elisabeth Quercia, Ernesto La Mattina, Charalampos Z. Patrikakis, Liz Bacon, Georgios Loukas, and Lachlan Mackinnon. *Serious Games: An Attractive Approach to Improve Awareness*, pages 1–9. Springer International Publishing, Springer Cham, 2018.

Exploring a Board Game to Improve Cloud Security Training in Industry

Tiange Zhao ✉ 

Siemens AG, Munich, Germany
Universität der Bundeswehr München, Germany

Tiago Espinha Gasiba ✉ 

Siemens AG, Munich, Germany
Universität der Bundeswehr München, Germany

Ulrike Lechner ✉ 

Universität der Bundeswehr München, Germany

Maria Pinto-Albuquerque ✉ 

University Institute of Lisbon (ISCTE-IUL), ISTAR, Portugal

Abstract

Nowadays, companies are increasingly using cloud-based platform for its convenience and flexibility. However, companies still need to protect their assets when deploying their infrastructure in the cloud. Over the last years, the number of cloud-specific vulnerabilities has been increasing. In this work, we introduce a serious game to help participants to understand the inherent risks, understand the different roles, and to encourage proactive defensive thinking. Our game includes an automated evaluator as a novel element. The players are invited to build defense plans and attack plans, which will be checked by the evaluator. We design the game and organize a trial-run in an industrial setting. Our preliminary results bring insight into the design of such a game, and constitute the first step in a research using design science.

2012 ACM Subject Classification Computer systems organization → Cloud computing; Social and professional topics → Computer and information systems training

Keywords and phrases cloud security, cloud control matrix, shared-responsibility model, industry, training, gamification

Digital Object Identifier 10.4230/OASIScs.ICPEEC.2021.11

Category Short Paper

Funding This work is partially financed by Portuguese national funds through FCT – Fundação para a Ciência e Tecnologia, I.P., under the projects FCT UIDB/04466/2020 and FCT UIDP/04466/2020. Furthermore, the fourth author thanks the Instituto Universitário de Lisboa and ISTAR, for their support.

1 Introduction

Cloud computing is a relevant and important architecture to provide IT services. The promised value of cloud service providers (CSP) is to provide high levels of service and security and save business costs. Significant players have dedicated themselves to provide cloud-based solutions to the customer. MindSphere [11] is an example of industrial Internet of Things (IIoT) as a service. It powers IoT solutions from the edge to the cloud. A cloud solution can also be found in Siemens' health care industry. Teamplay [15] is cloud-based network aimed to bring together health care professionals in a team effort. The number and volume of applications and systems that are deployed and maintained in the cloud increases.

Cloud-based systems are exposed to security threats as listed in [3] such as nefarious use of cloud services and lack of cloud security architecture and strategy. The current standards [2, 7] and guidelines [4, 1] on cloud security describe the roles and responsibility in cloud



© Tiange Zhao, Tiago Espinha Gasiba, Ulrike Lechner, and Maria Pinto-Albuquerque;
licensed under Creative Commons License CC-BY 4.0

Second International Computer Programming Education Conference (ICPEEC 2021).

Editors: Pedro Rangel Henriques, Filipe Portela, Ricardo Queirós, and Alberto Simões; Article No. 11; pp. 11:1–11:8

OpenAccess Series in Informatics



OASIS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

computing. These standards and guidelines are currently communicated to developers and managers largely by means of security training. It becomes imperative that a company helps its developer and manager understand the importance of cloud security and more precisely how it relates to daily work. Despite the many measures taken, cloud security awareness needs to be improved. We propose a serious game to address this challenge. Our research interest is to design a serious game to facilitate the training of developers and managers about cloud security, especially the roles and responsibilities and the collaboration between cloud service provider and customer. This work presents a preliminary result of a table top game prototype that is designed to introduce fundamental concepts in cloud security.

The research contribution of this work is to propose a possible serious game prototype to increase awareness in cloud security within industry setting and provides the result of game trial-run to verify the proposed prototype.

This article is structured in the following sections: section 2 introduces the related work in area of serious games in information security. Section 3 summarizes the method we use in our research. Section 4 describes the design of our game including the evaluator as a novel element in the prototype. Section 5 shares the feedback we collected from our trial run and our thinking upon it. Section 6 concludes this work and give an outlook into the future research direction.

2 Related Work

This article contributes to the understanding of the design of serious games to increase cloud security. The review of serious games in cyber-security by Shostack [14] demonstrates number and amount of serious games in the domain of cybersecurity.

Shostack presented in [13] a card game *Elevation of Privilege* that draws developers into threat modeling, whose importance used to be underestimated. By designing a cyber-physical systems game, Frey et al. [8] studied the information security field's decision-making process. Romand-Latapie pointed out in [12] that a role-playing game similar to *Dungeons and Dragons* was helpful in training neophyte audience to the basic principles of computer security. They might include cloud computing as a single element in the game design. Still, cloud computing's specific security topics are not addressed, such as shared responsibility model, cloud-specific threats and mitigation.

3 Method

Our research approach is guided by the design science paradigm according to Hevner et al. [9]. The method literature on design science describes the design of a useful artifact as a creative search process for a useful solution.

This paper presents a first step in the creative design process towards a serious game that raises awareness for cloud security among users of industrial cloud services. The design process includes various brainstorming sessions in which the topic "cloud security" and first ideas of a game and design principles of a game (cf. Sect. 4) were developed. The topic was chosen to be able to draw from professional experiences as well as game design and security expertise from the various authors of this paper. It was also chosen because of its relevance in industrial settings.

This first game has been designed in early 2021. A first game run with participants and observers with research interest and knowledge in security topics of cloud service provision was organized in February 2021. We collected the data anonymously and the participants

consented to take part in our study. The data collected in participatory observation was analysed, using as source the recorded material of the game session. This paper represents the state of the discussion after data analysis and reflection on next steps in the design process.

4 Game Design

The initial design elements and the game prototype are introduced in this section.

4.1 Initial design elements

In the first brainstorming sessions, the topic “Cloud Security” was identified and the format of the game: it should be a board game that can either be played face-to-face or as a virtual session to accommodate for the restrictions imposed by the COVID crisis. It was also determined that it should take into account the insider perspective of non-compliance with security policies.

By its nature, cloud security provides several elements that could be built into a board game: 1) Cloud security is a constant fight between defender and attacker. 2) For both defender and attacker, there are some constraints on resources. 3) Defenders might have different responsibilities determined by the role they play in cloud security. 4) Attackers could take cloud-specific attack actions to take down cloud assets. Using the elements above together, we can build a board game prototype that is geared to help trainees gain a better understanding on cloud security.

In the design phase, the following core features are identified. The game prototype aims to address those features:

Feature 1: Cloud Security Kill Chain. As Assante et al. mentioned in [5], Cyber attackers do not target systems in single incident and breach. Attackers in cloud security incidents plan the attack step by step. In the game, the attack element should consist of at least several phases instead of a “single shot”.

Feature 2: 100 percent security does not exist. Due to constraints on resources, in reality we never aim at 100 percent security in products and solutions. A correct countermeasure does not remove the threat completely. For example, implementing a strong password policy is considered an effective countermeasure to abusing credential. Strong password policy as a countermeasure does not eradicate the threat completely, just lowers the risk. Therefore the defending element should not guarantee any 100 percent security in the game.

Feature 3: Defense-in-depth helps. Use of defense-in-depth strategy, as Kuipers et al. advice [10], in general, improves protection against cyber-threats. Due to the uncertainty of attacker’s move, defense-in-depth strategy covers more possible attacks and should be encouraged in the game. The game prototype should address defense-in-depth strategy too.

4.2 Game Process

The game prototype requires a Game Master (GM), who organizes and hosts the game. Before the game starts, the GM explains the rules and the process to participants.

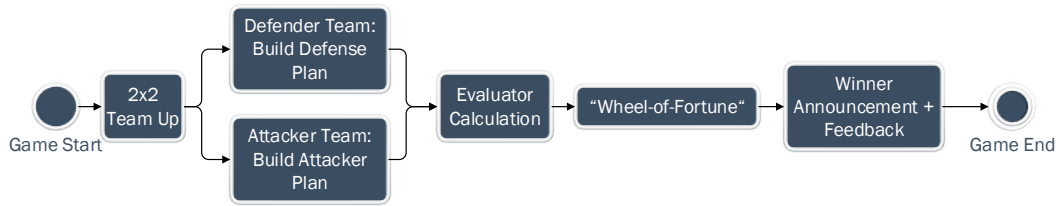
The idea of the game is that defender team develops a defense plan and the attacker team an attack plan. They use a board in which they place cards to model attack and defense plan. Attackers and defenders can only place a limited set of cards. This reflects that in reality neither attacker nor defender have unlimited resources and need to prioritize accordingly.

11:4 Exploring a Board Game to Improve Cloud Security Training in Industry

This development of attack and defense plans is done in teams. Teams use breakout rooms to discuss and develop plans.

There are in total 40 cards, of which 24 cards are made available to defender team and 16 to the attacker team, as table 1 shows. Each card of defender team is printed with one countermeasures to secure cloud assets, for example, “Network Segmentation” or “Information Encryption” (see figure 2). Each card of attacker team has one attack action on cloud assets, for example, “Network Service Discovery” or “Monitoring Escaping” (see figure 3).

Figure 1 presents the phases of the game as a flow chart.



■ **Figure 1** Game Process in Flowchart.

At the beginning of the game, the GM defines the two teams: *Defender* and *Attacker*. In the game, the task of the *Defender* team is to build a defense plan by assigning defense cards to the correct roles: cloud asset owner and cloud asset manager. They should decide to assign which 2 cards to Asset Owner and which 4 cards to Asset Manager. The defender team is not told which ones of the 24 cards belong to Cloud Asset Owner and which ones belong to Cloud Asset Manager. If a card is assigned to the wrong role, it will be excluded from the defense plan and have no effect in the evaluator. Table 1 shows specifically the number of cards of each category. There are two cards that could be assigned to both Asset Owner and Asset Manager. That explains why the sum of the second and fourth row of table 1 is 2 more than the total number of defense cards. Meantime, the task of the *Attacker* is to build an attack plan consisting of three steps: Gain Access, Launch Attack and Make Impact. They will get in total 16 attack cards categorized into the three steps and assign 2, 3 and 1 card(s) to the each step. Both teams have 20 minutes of time to build their defense plan and attack plan.

The attack cards, defense cards and the mapping between them are derived from MITRE ATT&CK [6] and the CSA cloud control matrix [4]. They are typical attack and defense actions in cloud environment.

The finished defense plan will be evaluated against the attack plan by an evaluator. The evaluator implements an algorithm to compute the probability according to which the winner of the game will be determined.

The GM explains the output of the evaluator to participants. For example, which defend card covers which attack card and which attack card is left without any countermeasure. The calculated output from the evaluator is a probability for the Defender Team to survive the attack in the end.

The next step is “Wheel-of-Fortune”. That is a virtual spinning wheel with different areas marked as “Attacker wins” or “Defender wins”. The area size is determined by the probability calculated in the previous step. In game run-time, “Wheel-of-Fortune” will be spun and the GM announces the winner.

As mentioned above, the evaluator is a novel element in the game process. It will be introduced in detail in the next sub-section.

■ **Table 1** No. of cards for defender team and attacker team.

Defender Team	Total no. of defense cards	24
	No. of defense cards belong to Asset Owner	8
	No. of defense cards to Asset Owner on Defense Plan	2
	No. of defense cards belong to Asset Manager	18
	No. of defense cards to Asset Manager on Defense Plan	4
Attacker Team	Total no. of attack cards	16
	No. of attack cards to chose from for Initial Access	5
	No. of attack cards for Initial Access on Attack Plan	2
	No. of attack cards to chose from for Launch Attack	8
	No. of attack cards for Launch Attack on Attack Plan	3
	No. of attack cards to chose from for Make Impact	3
	No. of attack cards for Make Impact	1

4.3 Evaluator

An evaluator is programmed to evaluate the defense plan against the attack plan through the calculation of the probability of the given defense plan to survive the attack. Throughout different game steps, it shows the reasoning of the final calculation given the defense plan choices against the attack plan. Finally, the evaluator outputs the calculated probability. There are some adjustable parameters for the evaluator such as the number of hints and the single success rate.

■ Number of Hints

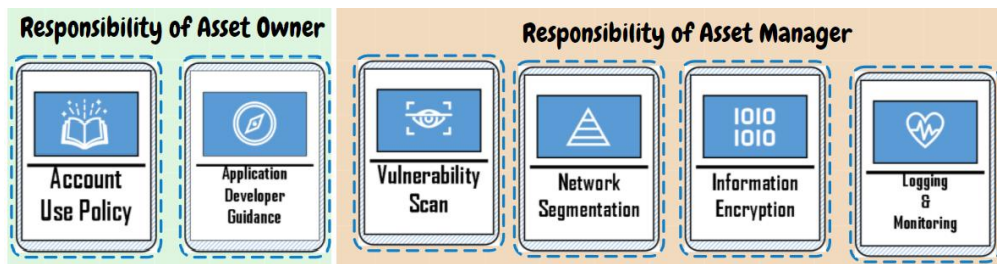
Hint is designed as an assistance for the defender team. It provides the defender team with a correct card assignment to a role in advance of the next game step. By assigning a correct card, it is guaranteed that the card will be included in the defense plan. Based on the observation of the game, the GM can decide whether to give up to two hints to the defender team.

■ Single Success Rate

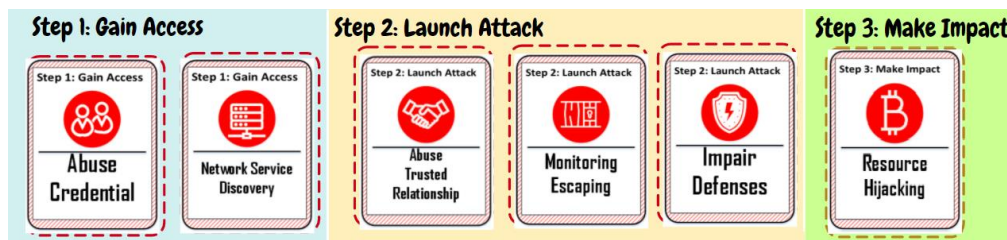
The single success rate is the likelihood of a single defense card successfully defend a mapped attack card. It is the same for all defense cards. It reflects the fact that there is no 100% security (Feature 2 in section 4). For example, we normally consider Multi-factor Authentication (MFA) as a valid countermeasure against account manipulation. When single success rate is set to 80%, it means in the game MFA has 80% chance of successfully defending account manipulation attack. In general, the higher the single success rate, the more likely the Defender Team will win. In the trial run, the single success rate is set to 80%.

5 Evaluation of the game

This section presents the result of the first trial run performed in February 2021 in an online format. Three players participated in the game. Two of them are experienced security expert and one of them is university student with basic cyber-security background knowledge. Two university professors were invited as observers. They observed the game without interfering. One of them joined the defender breakout room; the other joined the attacker breakout room.



■ Figure 2 Trial run: a screenshot of of the game board on the defender side.



■ Figure 3 Trial run: a screenshot of of the game board on the attacker side.

5.1 Trial Run Description

Participants were separated in two teams. Each team built their defense or attack plan on an online whiteboard application that was prepared with the cards and game board in advance. Figure 2 and figure 3 are the screenshots of the final defense and attacker plan in our game environment.

The defender team decided to let the asset owner enforce “Account Use Policy” to track information on login attempts and login time and provide “Application Developer Guidance” to minimize security weakness in the development phase. For asset manager, the defender team picked “Vulnerability Scan” to discover potential exploitable software; “Network Segmentation” to isolate critical systems, functions, or resources; “Information Encryption” to protect sensitive information and “Logging & Monitoring” to supervise the overall health and performance of cloud system. All the defense cards were assigned to correct roles, so no card was discarded. To gain access, the attacker team planned to use the card “Abuse Credential” to obtain credentials of existing account and use “Network Service Discovery” to get a list of potentially vulnerable services running on remote hosts. To launch attack, they planned to “Abuse Trusted Relationship” to breach through 3rd party provider; use “Monitoring Escaping” techniques to avoid detection and “Impair Defenses” to maliciously modifies components of a victim environment. In the last step, attacker team chose to make impact by “Resource Hijacking” to leverage the resources of co-opted systems to earn virtual currency.

At the end of the game, both plans were input into the evaluator. The evaluator calculated the final probability of the defense plan against the attack plan to be 96%. The probability was then set to the Wheel-of-Fortune. Despite the attacker team having only a 4% chance of winning, the outcome of the Wheel-of-Fortune was such that this team has won. This was a surprising event for all, but it reflects Feature 2 in section 4 that no defense is 100% secure.

5.2 Results and Discussion

After the trial run, players and observers were invited to exchange their opinions on the game through open discussions. Most of them agreed that the concept was helpful and the game itself was engaging. They also gave feedback on improvement. The following part summarizes the key information of their reviews.

The student participated in the trial run mentioned, *“I think it was pretty cool. It has some cyber-security notions that I still don’t really know, so I tried to see both the cheat-sheet and the task. Team environment helped.”* We implied from this piece of review that students or beginners can play the game with the prepared cheat-sheet and benefit by teaming up with experienced experts.

One of the security expert suggested, *“The number of rectangles you can assign to the roles represents the resource – you cannot do everything.”* When we design the game prototype, we considered that in reality, we could never implement all the countermeasures. They need to be considered individually and prioritized. That reflects the the defensive thinking in real world.

We also got feedback such as *“Why the attack ‘Impair Defenses’ is covered by the defense ‘Logging & Monitoring’ is not clear to me...”* As described in section 4, the evaluator checks the mapping between the chosen attack and defense cards. It shows the result in the run-time but does not provide any explanation yet. Further refinement of the evaluator’s algorithm and output should be planned for future work.

5.3 Conclusion on the Game Logic and Material

The goal of trial-run is to collect player feedback and draw new ideas for the next design iteration. As we can see from the feedback above, on one hand the participants find the game engaging and reflects pain points in building an effective defense for cloud asset in real world. On the other hand, the constructive feedback was collected and will be addressed in future work. From the designer perspective, the game logic was shown to be reasonable in the trial-run and the prepared material such as cheat-sheet, game board and cards have served their design purposes and helped the players in the game.

6 Conclusion and Future Work

Cloud solutions and deployments are becoming pervasive in the industry. In recent years, the rising number of cloud-related security incidents has shown that companies need to better protect their assets in the cloud. This work proposes a method to address this issue through a serious game, and is a first step in a research being conducted in the industry following the design science methodology. The game is designed based on MITRE ATT&CK [6] and CSA cloud control matrix [4] to reflect real-life situations encountered by companies deploying cloud environments. The goal of the game is to help its players to understand the different dangers that cloud systems face, increase players’ understanding of the responsibilities of different roles, and encourage proactive defensive thinking. Towards this goal, a preliminary design of the serious game is presented and is evaluated with real players from the industry in a trial run.

Valuable feedback was collected in the game that enables to steer the design research. To address the constructive feedback collected in trial-run regarding the evaluator, we would like to invite cloud security experts to review the defense and attack on our game card and the mapping between them as one of our next steps. Limitations on the number of players

and variations in the participants' background and experience are inherent to this industrial setting and preliminary study. In the future, we would like to invite more participants to join further trial runs and evaluate the level of cloud security awareness before and after playing the game.

References

- 1 Cloud Security Alliance. Security guidance for critical areas of focus in cloud computing v4.0, 2017. URL: <https://cloudsecurityalliance.org/artifacts/security-guidance-v4/>.
- 2 Cloud Security Alliance. Requirements for bodies providing star certification, 2020. URL: <https://cloudsecurityalliance.org/artifacts/requirements-for-bodies-providing-star-certification/>.
- 3 Cloud Security Alliance. Top threats to cloud computing: Egregious eleven deep dive, 2020. URL: <https://cloudsecurityalliance.org/artifacts/top-threats-egregious-11-deep-dive/>.
- 4 Cloud Security Alliance. Cloud controls matrix v4, 2021. URL: <https://cloudsecurityalliance.org/artifacts/cloud-controls-matrix-v4/>.
- 5 Michael J Assante and Robert M Lee. The industrial control system cyber kill chain. *SANS Institute InfoSec Reading Room*, 1, 2015.
- 6 MITRE ATT&CK. Tabletop security games & cards, 2020. URL: <https://attack.mitre.org/versions/v8/matrices/enterprise/cloud/>.
- 7 Carlo Di Giulio, Read Sprabery, Charles Kamhoua, Kevin Kwiat, Roy H Campbell, and Masooda N Bashir. Cloud standards in comparison: Are new security frameworks improving cloud security? In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pages 50–57. IEEE, 2017.
- 8 Sylvain Frey, Awais Rashid, Pauline Anthonysamy, Maria Pinto-Albuquerque, and Syed Asad Naqvi. The good, the bad and the ugly: a study of security decisions in a cyber-physical systems game. *IEEE Transactions on Software Engineering*, 45(5):521–536, 2017.
- 9 Alan Hevner, Salvatore March, and Jinsoo Park. Design science in information systems research. *Management Information Systems Quarterly*, 2004.
- 10 David Kuipers and Mark Fabro. Control systems cyber security: Defense in depth strategies. Technical report, Idaho National Laboratory (INL), 2006.
- 11 Dimitri Petrik and Georg Herzwurm. IoT ecosystem development through boundary resources: a Siemens MindSphere case study. In *Proceedings of the 2nd ACM SIGSOFT International Workshop on Software-Intensive Business: Start-ups, Platforms, and Ecosystems*, pages 1–6, 2019.
- 12 Tiphaine Romand-Latapie. The NeoSens training method: Computer security awareness for a neophyte audience, 2016. URL: <https://airbus-seclab.github.io/dnd/us-16-Romand-Latapie-Dungeons-Dragons-And-Security-wp.pdf>.
- 13 Adam Shostack. Elevation of privilege: Drawing developers into threat modeling. In *2014 {USENIX} Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, 2014.
- 14 Adam Shostack. Tabletop security games & cards, 2021. URL: <https://adam.shostack.org/games.html>.
- 15 Dina Simunic, Antun Kerner, and Srecko Gajovic. Digital mediators as key enablers of navigation toward health in knowledge landscapes. *Croatian medical journal*, 2018.

A System Architecture to Detect and Block Unwanted Wireless Signals in a Classroom

Daniel Barros  

Instituto Politécnico de Viana do Castelo, Portugal

Paulo Barros  

Instituto Politécnico de Viana do Castelo, Portugal

Emanuel Lomba  

Instituto Politécnico de Viana do Castelo, Portugal

Vítor Ferreira  

Instituto Politécnico de Viana do Castelo, Portugal

Pedro Pinto  

Instituto Politécnico de Viana do Castelo, Portugal

ISMAI and INESC TEC, Porto

Abstract

The actual learning process in a school, college or university should take full advantage of the digital transformation. Computers, mobile phones, tablets or other electronic devices can be used in learning environments to improve learning experience and students performance. However, in a university campus, there are some activities where the use of connected devices, might be discouraged or even forbidden. Students should be discouraged to use their own devices in classes where they may become alienated or when their devices may cause any disturbance. Ultimately, their own devices should be forbidden in activities such as closed-book exams. This paper proposes a system architecture to detect or block unwanted wireless signals by students' mobile phones in a classroom. This architecture focuses on specific wireless signals from Wi-Fi and Bluetooth interfaces, and it is based on Software-Defined Radio (SDR) modules and a set of antennas with two configuration modes: detection mode and blocking mode. When in the detection mode, the architecture processes signals from the antennas, detects if there is any signal from Wi-Fi or Bluetooth interfaces and infers a position of the unwanted mobile device. In the blocking mode, the architecture generates noise in the same frequency range of Wi-Fi or Bluetooth interfaces, blocking any possible connection. The proposed architecture is designed to be used by professors to detect or block unwanted wireless signals from student devices when supervising closed-book exams, during specific periods of time.

2012 ACM Subject Classification Computer systems organization → Architectures

Keywords and phrases campus, classroom, closed-book exam, fraud, wireless, detection, blocking, Software-Defined Radio

Digital Object Identifier 10.4230/OASICS.ICPEEC.2021.12

Category Short Paper

1 Introduction

The last decade has seen an increase in the use of personal mobile devices such as smartphones [12]. In the context of a campus environment, such devices offer several advantages as they can be used to enhance the learning experience and students performance. However, according to the survey in [16], 95% of students bring their own mobile phones to class every day, challenging the Bring Your Own Device (BYOD) policy and generating other concerns. When students use their own devices in some activities, they may become alienated, lose attention in class, and suffer from social isolation, powerlessness, meaninglessness [10]. The



© Daniel Barros, Paulo Barros, Emanuel Lomba, Vítor Ferreira, and Pedro Pinto; licensed under Creative Commons License CC-BY 4.0

Second International Computer Programming Education Conference (ICPEEC 2021).

Editors: Pedro Rangel Henriques, Filipe Portela, Ricardo Queirós, and Alberto Simões; Article No. 12; pp. 12:1–12:7

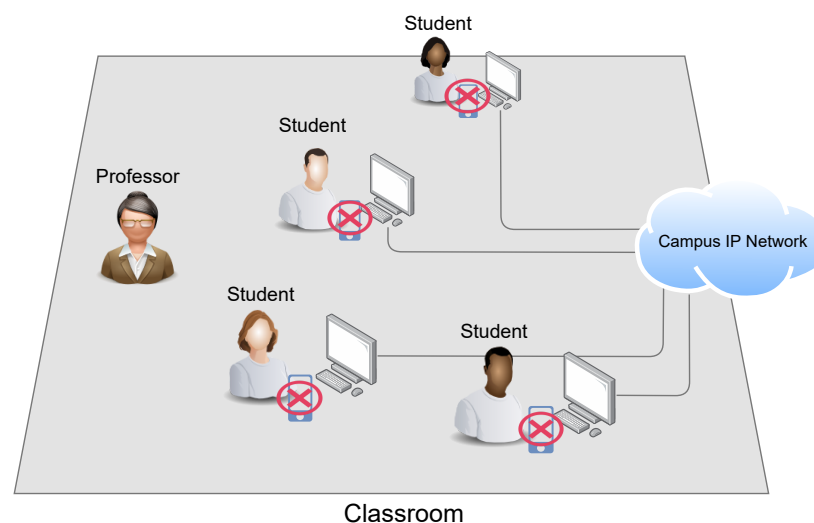
OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

12:2 System Architecture to Detect and Block Wireless Signals

classes and other events in campus can also be disturbed by the use of such devices, due to recurrent buzzing as students receive or send messages [6]. Last but not least, mobile devices can be used for exam fraud, *i.e.* students may use these devices to exchange with colleagues, information such as photographs or texts about the exam subject, during examinations [1]. Fig. 1 presents a typical scenario in a college or university where a classroom is equipped with computers/workstations connected to the campus IP network. In this classroom, on a specific time period, a group of students is about to perform a task, *e.g.* an online closed-book exam, supervised by a professor. Each student uses the college or university computer but may also bring to the classroom their own mobile phones (and/or other connected devices such as tablets, ear buds, etc). Since it is a closed-book exam, to prevent the misuse of student'



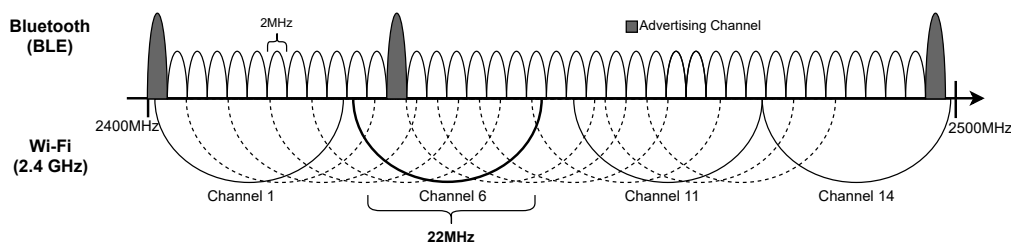
■ **Figure 1** Classroom scenario.

communication devices, the professor would require the students to turn off their phones or to put them in “Airplane Mode”. However, not all students comply with this rule. In this scenario, the students’ mobile devices may use their Wi-Fi and Bluetooth interfaces. The detection of signals in the classroom environment, originated in the devices’ transceivers is of great importance to detect eventual abuses and locate the abusing device. Also, blocking communications from/to these devices can be the last mile resource to assure that everyone complies with the rules of the exam. This paper presents an architecture to detect and block unwanted wireless signals in the typical scenario described. This architecture is expected to aid professors detecting non allowed devices using Wi-Fi or Bluetooth during activities such as a closed-book exam in a classroom. This paper is organized as follows. Section 2 presents the background and related work. Section 3 details the proposed architecture and discusses its implementation. Section 4 draws conclusions.

2 Background and Related Work

In a campus environment, the BYOD policy may affect infrastructure security levels [5], academic fairness and students performance. A study in [16] refers that 95% of students bring their phones to class every day, 92% use their phones to text messages during class

time, and 10% of the students admit to have texted during an exam on at least one occasion. In [14] the author highlights that young adults, such as university students with higher social media interactions, seem to feel more socially isolated than their counterparts with lower social media interaction [14], resulting in a deficit of student performance. In [7] the author surveyed a group of students to evaluate academic dishonesty, using the same anti-social behaviour pattern in [11]. The results showed that, as the students grew older, the prevailing of cheating grew exponentially higher, jumping from 10% in the age of 7, to 65-70% in the ages from 17 to 18 years old. The percentage of students that cheated multiple times hovered around 15 to 20% for students from 17 to 18 years old. The study in [1] highlights that the use of technology-facilitated learning methods is more common on college majors for their expertise and financial affluence, as well as majors that require the use of technologies are also more likely to involve technology in the cheating process. The devices used by the students in campus have mainly 3 types of interfaces to connect to the “external world”: the cellular communications, Bluetooth and Wi-Fi. Cellular or mobile communications can be used in compliance with 2G standards (GSM&GPRS), operating at 900MHz, 3G standards (UMTS/IMT, IMT-2000, CDMA2000), or 4G standards (LTE), operating at 1800MHz, 2100MHz, and 2600MHz. In each country or region, cellular communications must comply with strict regulations, require specific licenses to operate, and must provide particular service availability, therefore, they are not considered in this study. Bluetooth is used to connect the device to wireless peripherals such as earphones or keyboards. It operates in a range of frequencies from 2402 to 2480MHz, uses channels of 2MHz and has a range from 4.5 to 100 meters depending on the Bluetooth version. Regarding Wi-Fi, the latest versions of its standards [8] include IEEE 802.11g and IEEE 802.11n, which operate in frequency ranges from 2401 to 2495MHz using 14 channels of approximately 20MHz, and IEEE 802.11ac, which operates only at the 5GHz band. The current proposal does not contemplate this band, since it would require different hardware. Fig. 2 resumes the spectrum used by Bluetooth Low Energy (BLE) and Wi-Fi 2.4GHz standards. In order



■ **Figure 2** Spectrum utilization of Bluetooth and Wi-Fi standards.

to detect wireless signals, the frequency spectrum on frequencies used can be analysed. In [13] device detection is achieved through the periodic detection of the probe signal, used to recognize nearby networks, of Wi-Fi enabled devices. Other strategies perform passive monitoring on the network as in [4] where the author presents a solution to prevent Wi-Fi misuse by students, by checking the power of wireless signal received and controlling the network access, through passive traffic sniffing. This solution, implemented in Iowa State University Campus, allows an instructor to filter traffic, based on certain requirements. To block wireless signals, a signal blocker, also known as jammer or inhibitor can be used. A signal blocker is a device that generates a static signal or random noise over a single or a range of frequencies, at such a power level that a nearby device cannot use that spectrum anymore thus, blocking the normal operation of communications on that device. Regarding signals

blocking techniques, in [2] the authors assess the performance of Wi-Fi under the impact of jammers and present a reactive jammer that adjusts its jamming strategy according to the states of participating nodes. In [9] the authors present a comparative analysis between sweeping and non-sweeping frequency jamming, concluding that both techniques present no differences regarding jamming effectiveness, although sweeping frequency jamming is taken as more flexible. Authors in [18] describe low-layer attacks against Wi-Fi networks that can be performed using modified firmware and an inexpensive, “off-the-shelf Wi-Fi dongle”. In [15] the authors assess the impact of jamming in a wireless network by varying the power and frequency of transmission along with the payload size at sender and analysing the impact on packet throughput. In [3] authors perform selective jamming of BLE advertising packages by using a low-cost, small-sized, and power efficient implementation. In order to detect or block a signal, a Software-Defined Radio (SDR) module can be used. An SDR module is a device that receives and digitizes a signal and implements by software most of the traditional signal processing components such as filters, mixers, modulators, or detectors [17]. In the opposite direction, an SDR module working as a transmitter, converts a digital input into its equivalent radio signal. These approaches enable dynamic (re)configurations of the radio, thus broadening the range of applications of their hardware. Depending on its hardware characteristics, a receiving SDR module can operate in a standalone fashion or be connected to a computer that would be in charge of processing the base-band signal and extracting any useful information. Table 1 presents common and low-cost SDR modules with relevant characteristics for the current context.

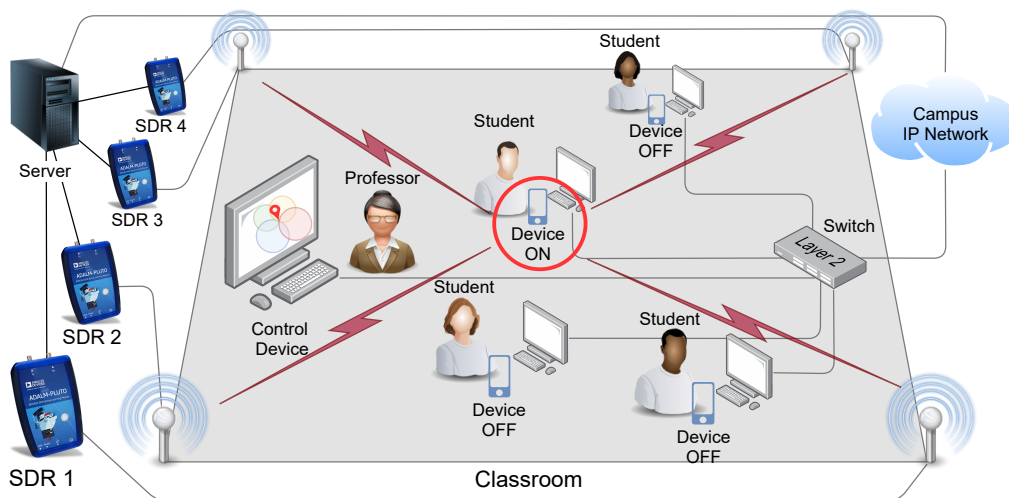
■ **Table 1** Characteristics of surveyed SDR modules.

	ADALM-Pluto ¹	HACK-RF One	LimeSDR	LimeSDR mini	BladeRF x40	BladeRF 2.0 micro xA4
Price	150~200€	250~300€	650~700€	300~350€	500~550€	550~600€
RX-TX capability	RX & TX	RX & TX	6RX & 4TX	RX & TX	RX & TX	2RX & 2TX
Complexity	Full-Duplex	Half-Duplex	Full-Duplex	Full-Duplex	Full-Duplex	Full-Duplex
TX power (dBm)	5.7	15	10	-	6	8
Freq. Range (MHz)	70 - 6000	1 - 6000	0.1 - 3800	10 - 3500	300 - 3800	47 - 6000
Bandwidth (MHz)	56	20	61.44	30.72	40	61.44
Interface	USB 2.0	USB 2.0	USB 3.0	USB 3.0	USB 3.0	USB 3.0

1) Expanded version (bandwidth, frequency range and dual core CPU).

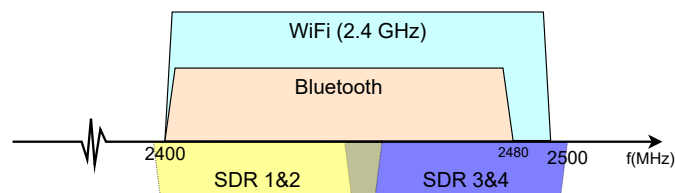
3 Detection and Blocking Architecture Proposal

Fig. 3 presents the proposed architecture to detect and block wireless signals deployed on a scenario such as the one described in the Fig. 1. The scenario is composed of a classroom of around 100m², where the students take a closed-book exam. In this case, all students except one (in the center of the figure) have their mobile phones disconnected from Wi-Fi and Bluetooth. The proposed architecture is composed of a control device, a server, and a set of SDR modules with their antennas. The control device is a common computer where the professor, via a Web interface, configures the system settings, monitors the execution of the exam, and may check the location of a not-allowed connected device. The server is a computer that hosts an Web server, a back-end script, and a database, in order to store configurations and provide the necessary Application Programming Interface (API) to the control device and for the SDR modules. The server receives the settings of the control device and configures the SDR modules for detection or blocking mode. The SDR modules are controlled to receive or block the Bluetooth or Wi-Fi signals in the 2.4GHz band and



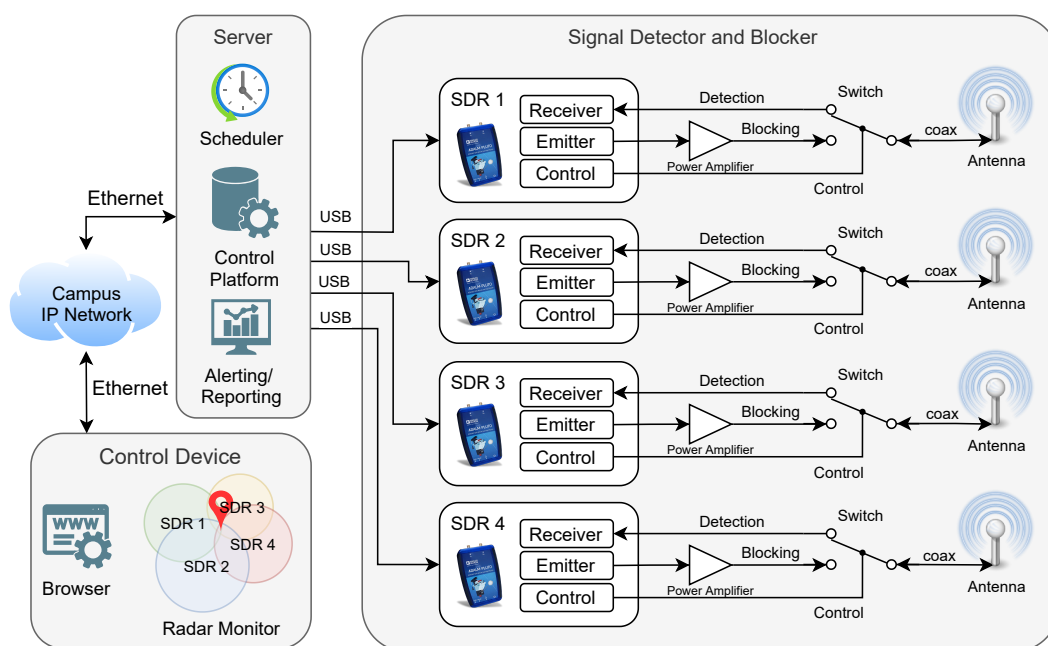
■ **Figure 3** Proposed architecture to detect and block wireless signals.

thus, from the set of modules presented in Table 1, the SDR module ADALM-Pluto was chosen as the most cost efficient solution. This model is able to work as a detector and as a blocker, but has a bandwidth limit of 56MHz. This means that, in the current scenario, the SDR modules should work in pairs, allowing them to cover the entire 2.4GHz Wi-Fi band (of around 100MHz from 2400 to 2500MHz), as presented in Fig. 4. A detailed schematic of the



■ **Figure 4** Wi-Fi and Bluetooth signals covered by two pairs of SDR modules.

proposed architecture is presented in Fig. 5. The professor accesses the system configuration interface in a browser on the Control Device, which is connected via Ethernet to the campus IP network and the Server. The Server connects to each SDR module via USB and each SDR is connected to an antenna using a coaxial cable. In case the system is configured by the professor in the Detection Mode, each SDR module is configured by the Server to work as a detector, commuting the switch to the Detection mode and enabling the Receiver block. The Server processes the signals captured by each SDR and will detect if there is any unauthorised signal from/to Wi-Fi and Bluetooth interfaces (cf. Fig. 4) in the classroom, and infer its respective device location. The detection result is presented to the professor using the Radio Monitor in the Control Device. In case the system is configured by the professor in the Blocking Mode, each SDR module is configured by the Server to work as a jammer, positioning the switch in the Blocking Mode and enabling the Emitter block in each SDR module. Each SDR module will generate a noise signal that will be amplified by the power amplifier in the same range of the Wi-Fi and Bluetooth signals (cf. Fig. 4), thus cancelling any transmission possibility. Receiving or transmitting in specific frequencies may have legal implications. Thus, the proposed architecture should comply with all the legal requirements inside campus, national (Portuguese) and European Union laws.



■ Figure 5 System Architecture.

4 Conclusions and Future Work

Nowadays, students have their own mobile devices and bring them to classes in colleges and universities, taking advantage of the BYOD policy. Although normally allowed in some learning scenarios where students are intended to be with maximum concentration, the use of such devices is not recommended, since it may lead to problems such as total alienation and classes disturbance. There are also cases where these devices are definitely forbidden, such as the case of closed-book exams. To limit the use of mobile devices during specific time frames in a particular classroom it is useful to detect and block the signals from/to these devices. This article presents a system architecture to detect and block wireless signals from Wi-Fi or Bluetooth interfaces. When in detecting mode, this architecture includes SDR modules to detect and infer location of a device transmitting/receiving unwanted signals. The proposed architecture is expected to aid professors detecting non allowed devices in a classroom in scenarios such as a closed-book exam. Further efforts will be applied to build and test a prototype of the proposed architecture. Future work should also address the legal implications and provide results of technology acceptance.

References

- 1 Eric M. Anderman and Tamera B. Murdock. *Psychology of Academic Cheating*. Elsevier Inc., 2007. doi:10.1016/B978-0-12-372541-7.X5000-1.
- 2 Emrah Bayraktaroglu, Christopher King, Xin Liu, Guevara Noubir, Rajmohan Rajaraman, and Bishal Thapa. Performance of IEEE 802.11 under jamming. *Mobile Networks and Applications*, 18(5):678–696, October 2013. doi:10.1007/s11036-011-0340-4.
- 3 Sebastian Bräuer, Anatolij Zubow, Sven Zehl, Mehran Roshandel, and Soroush Mashhadi-Sohi. On practical selective jamming of Bluetooth Low Energy advertising. In *2016 IEEE Conference on Standards for Communications and Networking, CSCN 2016*. Institute of Electrical and Electronics Engineers Inc., December 2016. doi:10.1109/CSCN.2016.7785169.

- 4 Andrew Daniel Buschbom and Andrew Daniel Buschbom. Restricting wireless network access within the classroom. *Iowa State University Capstones Theses And Dissertations*, 1(1):1–47, January 2007. doi:10.31274/rtd-180813-16155.
- 5 Paulo Costa, Ricardo Montenegro, Teresa Pereira, and Pedro Pinto. The Security Challenges Emerging from the Technological Developments. *Mobile Networks and Applications*, 24(6):2032–2037, 2019. doi:10.1007/s11036-018-01208-0.
- 6 Renata Adams Fernandes, Deisi Cristina Gollo Marques Vidor, and Alcyr Alves de Oliveira. The effect of noise on attention and performance in reading and writing tasks. In *CoDAS*, volume 31-4. SciELO Brasil, 2019.
- 7 António Castro Fonseca. Desonestidade nos trabalhos escolares: Dados de um estudo português. *Revista Portuguesa de Pedagogia*, 43(2):107–124, July 2009. doi:10.14195/1647-8614_43-2_7.
- 8 IEEE802. IEEE 802.11, The Working Group Setting the Standards for Wireless LANs, 2015. URL: <https://www.ieee802.org/11/>.
- 9 Zhang Jie, Wu Gang, and Yan Xiaowei. Comparative Analysis of Sweeping Frequency Jamming and Non-sweeping Frequency Jamming in Partial-band Jamming Based on Projectile-carried Communication Jamming. In *International Conference on Communication Technology Proceedings, ICCT*, volume 2020-Octob, pages 379–383. Institute of Electrical and Electronics Engineers Inc., October 2020. doi:10.1109/ICCT50939.2020.9295755.
- 10 Genevieve Marie Johnson. Student alienation, academic achievement, and webct use. *Journal of Educational Technology & Society*, 8(2):179–189, 2005.
- 11 Rolf Loeber, Jeffrey D. Burke, and Dustin A. Pardini. Development and etiology of disruptive and delinquent behavior, April 2009. doi:10.1146/annurev.clinpsy.032408.153631.
- 12 S. O’Dea. Smartphone users 2020 | Statista, 2020. URL: <https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/>.
- 13 L. Oliveira, D. Schneider, J. De Souza, and W. Shen. Mobile device detection through wifi probe request analysis. *IEEE Access*, 7:98579–98588, 2019. doi:10.1109/ACCESS.2019.2925406.
- 14 Brian A. Primack, Ariel Shensa, Jaime E. Sidani, Erin O. Whaite, Liu yi Lin, Daniel Rosen, Jason B. Colditz, Ana Radovic, and Elizabeth Miller. Social Media Use and Perceived Social Isolation Among Young Adults in the U.S. *American Journal of Preventive Medicine*, 53(1):1–8, July 2017. doi:10.1016/j.amepre.2017.01.010.
- 15 Sethuraman Rao, S. Deepak, and Preeja Pradeep. Parametric analysis of impact of jamming in wireless sensor networks. In *IFIP International Conference on Wireless and Optical Communications Networks, WOCN*, 2013. doi:10.1109/WOCN.2013.6616191.
- 16 Deborah R Tindell and Robert W Bohlander. The use and abuse of cell phones and text messaging in the classroom: A survey of college students. *College Teaching*, 60(1):1–9, 2012.
- 17 Walter HW Tuttlebee. *Software defined radio: enabling technologies*. John Wiley & Sons, 2003.
- 18 Mathy Vanhoef and Frank Piessens. Advanced Wi-Fi attacks using commodity hardware. In *ACM International Conference Proceeding Series*, volume 2014-December, pages 256–265. Association for Computing Machinery, December 2014. doi:10.1145/2664243.2664260.

A Teaching Assistant for the C Language

Rui C. Mendes   

Centro Algoritmi, Departamento de Informática, University of Minho, Braga, Portugal

José João Almeida   

Centro Algoritmi, Departamento de Informática, University of Minho, Braga, Portugal

Abstract

We introduce a C tutor that can help instructors manage classes with many students learning to program in C. Nowadays, it is easy to evaluate code but it is hard to provide good feedback. We introduce a tool to help instructors provide students with feedback concerning their implementation and documentation for honing their programming skills. This tool is implemented in Python and is available at https://github.com/rcm/C_teaching_assistant.

2012 ACM Subject Classification Applied computing → Computer-assisted instruction; Software and its engineering → Empirical software validation

Keywords and phrases Software metrics, Documentation extractor, Domain specific language, Query language, Report generation

Digital Object Identifier 10.4230/OASISs.ICPEEC.2021.13

Category Short Paper

Supplementary Material *Software:* https://github.com/rcm/C_teaching_assistant

Funding This research has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020.

1 Introduction

Learning to program is a difficult task [8]. There are many concerns involved when teaching the first imperative programming language. Students are often fairly disorganized and have difficulty writing and understanding code. The common difficulty involved in learning to program is that students often write code tentatively by applying several patches until it finally is able to solve the given task. However, when involved in a project, this *ad hoc* methodology is quite detrimental. It is important to provide feedback concerning implementation. This means that students should realize which functions implemented are poorly written and should be improved.

When teaching classes with many students, it is hard to have enough teaching assistants for accompanying them and providing feedback on all fronts: writing code that implements the required specifications, is well documented and is easy to understand and maintain. In order to help instructors provide feedback to students, it makes sense to follow an automatic assessment strategy based on three pillars: functionality, readability and documentation.

The goal of the tool presented in this work is to offer a solution that will help instructors provide automatic assessment on two of these pillars: program readability/maintainability and documentation. With it, instructors will be able to produce reports that will provide feedback and help students fulfill these tasks. This system may also be used for summative assessment.

2 Software metrics

Using a tool capable of testing programs helps instructors understand whether the students were able to implement a solution capable of solving a given problem. This may be done by using many different online judge systems capable of automatically evaluating source



© Rui C. Mendes and José João Almeida;
licensed under Creative Commons License CC-BY 4.0

Second International Computer Programming Education Conference (ICPEEC 2021).

Editors: Pedro Rangel Henriques, Filipe Portela, Ricardo Queirós, and Alberto Simões; Article No. 13; pp. 13:1–13:8

OpenAccess Series in Informatics



OASIS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

code [10]. However, it is possible to create a program that solves a task while writing code that is hard to read and maintain. In fact, this could be detrimental because it would give students the wrong feedback: as long as it solves the problem, the solution is accepted.

There is a known relationship between code that is difficult to understand and software metrics. This was also reported when creating questions for exams [5]. Some studies suggest that it is important to include software metrics in feedback given to students [2] and have reported including these metrics in Fuzzy rules used for giving advice to students [4]. Both of these studies have used these metrics and incorporated them when evaluating small programming assignments for providing feedback. Our aim is to extend these studies by providing a tool in order to automatically collect these statistics over all functions in a project.

Given all these clues from literature, it makes sense to incorporate information about software metrics when evaluating programming projects. Thus, the authors decided to include some measures like lines of code, cyclomatic complexity [7] or software maintainability index [3] in summative assessment. These were incorporated with other concepts like not defining code in include files, not having warnings and not having global variables. However, when presenting these measures to students, they reported difficulties in understanding how to improve their program in order to improve their evaluation. One of the difficulties reported was in identifying the functions that were responsible for their low results.

One way of helping both instructors and students would be to create a report that details these measures on a per function basis. If students are able to peruse a table that details this information for each function, they will have further clues as to which functions are more complex and should be rewritten. Thus, they can try to refactor the code to address these issues and better understand these concepts. Even though there are tools readily available that can help gather these software metrics [11], they are not easy to use and only provide feedback per source code file. We aim to provide a tool that helps instructors evaluate students and/or create detailed reports using a large number of software metrics over all functions in a project.

3 Concerning documentation

The Bloom taxonomy is well known and can be applied when teaching computer science [9]. The cognitive categories involved in the taxonomy are to remember, understand, apply, analyse, evaluate and create. Some of these concepts can be applied when producing documentation. This task helps students understand and evaluate their code. While they write the documentation, they have to answer some key questions like how to explain what each function does, what it returns, what is the importance of each of the arguments it receives and what is the program flow necessary to implement the functionalities needed by their project. While producing code helps students to apply the knowledge they learned and use it to create a solution for a problem, documentation helps students analyse and evaluate their solution because it forces them to better understand how it works.

Evaluating documentation quality in a project is a cumbersome task because it involves checking that all the concepts in the program were explained: macros, types, functions and their arguments and return values. While a documentation coverage tool does not help instructors understand if the documentation that was produced is clear and well written, it can reduce the burden drastically. As far as the authors know, the automatic evaluation of documentation available when using `Doxygen` is `coverxygen` [6] that checks for documentation coverage. There are some common mistakes concerning student documentation. They usually:

- Forget to document some of the arguments;
- Copy information from other functions and thus document non-existing arguments;

- Forget to update the documentation after changing the function (e.g., adding or removing arguments);
- Document other entities as function arguments, like local variables or constants;
- State that a `void` function returns something.

Many of these mistakes can be checked automatically but there are no tools that help in this task. Coverxygen only gives information about documentation coverage, including datatypes, macros and function coverage but not argument coverage. Thus, while it is possible to use it to check whether all functions were documented, it does not help to identify the problems mentioned above.

4 The C Teaching Assistant

We introduce a tool, the C Teaching Assistant, that helps instructors produce documents that report what problems exist with the students' code or documentation. This tool analyses the users' submissions and gathers information about all the functions they contain. All the fields that can be used on the reports are presented in appendix A.

The tool is invoked by passing it several arguments that represent pathnames containing C code. Each of the arguments passed corresponds to a different project.

```
$ teaching_assistant.py PL[1-9]G[0-9]*
Enter query >
```

The tool recursively searches through all the existing C files (both header files and code files) and creates a table concerning all the information gathered. After consulting all the information, the user can use a domain specific language (DSL) to query the system. The DSL uses a syntax similar to SQL and is described in Listing 1. The queries can either be executed interactively or the resulting table may be converted into many formats including most markdowns, \LaTeX or HTML.

■ **Listing 1** The description of the query language. `expression` can contain arbitrary Python code and use field names.

```
SHOW <expressions separated by spaces or semicolons>
[HEADER <fields separated by spaces>]
[COND <conditions using fields and Python operands and functions>]
[SORT <fields separated by spaces>]
[COLOR <field> : <expression>[; <field> : <expression>]*]
[
GROUP_BY <fields separated by spaces>
[AGGREG <expressions separated by spaces or semicolons>]
]
```

5 Some examples of queries

Listing 2 shows an example that uses some of the measures collected by `Multimetric`, applies some functions to the values presented and uses a color code for signaling when these measures have non recommended values. When using something that returns a Boolean value, the system uses the green and red colors; when using a real number between zero and one, the system uses a palette. It is possible to use Python functions that return one color based on the value that was passed as well. The functions `scale_lower` and `scale_upper` will create functions that scale the values to the $[0, 1]$ interval using the lower and upper

13:4 A Teaching Assistant for the C Language

bounds. The function `scale_upper` will create a scaler that returns 0 for values lower or equal than the lower bound, 1 for values greater or equal than the upper bound and uses interpolation between these bounds. The function `scale_lower` is reciprocal. This example also sorts the table results by decreasing values of `maintainability_index` and increasing values of `cyclomatic_complexity` and `name`. Figure 1 shows a snippet of what can be seen on the terminal. While it is possible to produce the output with less decimal places, it would involve a more complicated query.

■ **Listing 2** Functions and some software engineering measures color coded by their severity. The query language allows the use of semicolons instead of spaces to be able to break lines in order for the query to be more readable.

```
SHOW name; cyclomatic_complexity; maintainability_index;
  scale_lower(1,10)(cyclomatic_complexity);
  scale_upper(60,100)(maintainability_index);
  0.5*scale_lower(1,10)(cyclomatic_complexity) +
  0.5*scale_upper(60,100)(maintainability_index)
HEADER name complexity maint_idx cmlpxty_grd maint_grd assessment
SORT -maintainability_index cyclomatic_complexity name
COLOR
  complexity : scale_lower(1,10)(complexity);
  maint_idx : maint_idx > 80; cmlpxty_grd : cmlpxty_grd;
  maint_grd : maint_grd; assessment : assessment
```

Listing 3 shows two examples of queries containing aggregation. These can use any Python function over fields and, in the current implementation of this tool, uses some functions declared in the `statistics` module. The first example reports some statistics concerning the number of effective lines of code for each project and function return type. The second example presents the mean number of useful lines of code and counts the number of functions with less than 10 lines of code for all projects. This example is not very readable but is there to illustrate the reason why the `AGGREGATE` keyword can receive tokens separated by semicolons and that it can use arbitrary Python code.

■ **Listing 3** Examples using aggregation.

```
SHOW name project return loc
GROUP_BY project return
AGGREG len(name) min(loc) mean(loc) max(loc)

SHOW name project return loc
GROUP_BY project
AGGREG mean(loc); (lambda L: len([x for x in L if x < 10]))(loc)
```

The system can also be used inside other programs. It is just a matter of first using the function `extract_all_functions` with a project and subsequently calling either the lower level function `function_query` that returns a list of lists containing the table or using the function `query` and pass it the structure returned from `extract_all_functions` and a string containing the query as illustrated in the examples given above (cf. Listing 4). Function `query` can also be passed an optional argument `fmt` that specifies which format (used by the module `tabulate` [1]) to use for the table.

■ **Listing 4** Using inside other Python scripts.

```
from teaching_assistant import *
info = extract_all_functions("/home/rui/repos/PL2G01")
```

```

result = query(info, """SHOW name return args
SORT name""")
print(result)
result = query(info, ["SHOW name project","SORT name"], fmt = "html")

```

Listing 5 shows an example of generating a report. In order to use this functionality, the command is executed with the `-t` keyword.

```
teaching_assistant -t example_report PL[1-9]G[0-9]*
```

■ **Listing 5** Example of generating a report.

```

# List of problematic functions
These functions are too complex and should be rewritten:

'''
SHOW project name cyclomatic_complexity maintainability_index
COND cyclomatic_complexity > 10 or maintainability_index < 80
SORT project -maintainability_index cyclomatic_complexity name
COLOR
    cyclomatic_complexity : scale_lower(0,50)(cyclomatic_complexity);
    maintainability_index : scale_upper(0,100)(maintainability_index)
'''

# Bad documentation
The following functions have no documentation:

'''
SHOW name
COND not comment
'''

These functions have a *@return* keyword for *void* functions:

'''
SHOW name comment
COND comment and return == 'void' and '@returns' in comment
'''

The following functions have problems with the definition of
their arguments. They either:
- Forget to document some of the function arguments, or
- Document things that are not function arguments

'''python
def arg_doc_problems(args, comment):
    actual = set(args.keys())
    reported = set(re.findall(r"@param\s+(\S+)", comment, re.M))
    return actual != reported
def arg_names(args):
    return list(args.keys())
'''

'''
SHOW name filetype filename arg_names(args) documented(comment)
COND arg_doc_problems(args,comment)
'''

```

name	complexity	maint_idx	cmplxty_grd	maint_grd	assessment
is_empty	0	100	1	1.0	1.0
penultimo	0	100	1	1.0	1.0
pop	0	100	1	1.0	1.0
top	0	100	1	1.0	1.0
has_type	1	100	1.0	1.0	1.0
main	1	100	1.0	1.0	1.0
duplica	2	100	0.8888888888888888	1.0	0.9444444444444444
enesimo	2	100	0.8888888888888888	1.0	0.9444444444444444
lex_linha	2	100	0.8888888888888888	1.0	0.9444444444444444
new_stack	2	100	0.8888888888888888	1.0	0.9444444444444444
popP	2	100	0.8888888888888888	1.0	0.9444444444444444
rodadres	2	100	0.8888888888888888	1.0	0.9444444444444444
trocadois	2	100	0.8888888888888888	1.0	0.9444444444444444
copian	3	100	0.7777777777777778	1.0	0.8888888888888888
parsee2	3	100	0.7777777777777778	1.0	0.8888888888888888
verifica_carater	3	100	0.7777777777777778	1.0	0.8888888888888888
nott	5	100	0.5555555555555556	1.0	0.7777777777777778
push	3	99.92049769252367	0.7777777777777778	0.9980124423130917	0.8878951100454348
parsee	3	97.99452174359152	0.7777777777777778	0.949863043589788	0.8638204106837829
converte_para_char	5	96.52391934110824	0.5555555555555556	0.913097983527706	0.7343267695416308
converte_para_double	6	96.1542362975608	0.4444444444444444	0.90385590743902	0.6741501759417322
converte_para_long	6	96.07022845886661	0.4444444444444444	0.9017557114716652	0.6731000779580548
decrementa	8	91.56801924883034	0.2222222222222222	0.7892004812207585	0.5057113517214904
incrementa	8	89.06398158095722	0.2222222222222222	0.7265995395239304	0.4744108808730763
print_stack	3	84.6730866972618	0.7777777777777778	0.6168271674315449	0.6973024726046613
xorx	10	81.5303145267608	0.0	0.5382578631690201	0.26912893158451007
e	10	80.66933794117628	0.0	0.5167334485294071	0.25836672426470353
ou	10	79.94165314864951	0.0	0.4985413287162377	0.24927066435811884
modulo	6	78.20989263681746	0.4444444444444444	0.45524731592043644	0.4498458801824404
soma	19	63.62646972307609	0.0	0.09066174307690229	0.045330871538451147
multiplica	20	62.87710754667034	0.0	0.07192768866675844	0.0359384443337922
expoente	20	62.54920051887836	0.0	0.06373001297195895	0.0318650648597947
subtrai	20	62.492758972679226	0.0	0.062318974316980656	0.031159487158490328
dividir	11	61.40891224648385	0.0	0.035222806162096276	0.0176114038081048138
tokenizador	51	47.186473217293056	0.0	0.0	0.0

■ **Figure 1** Result of running query on listing 2 on a given project.

The example is more or less self explanatory. It starts by creating a table of the functions that are too complex. Then it displays a table of the functions that don't have documentation. It shows the name of the functions and the condition uses a `Python` statement that checks if the string containing the comment is empty. Then it checks for functions that erroneously document the return value for `void` functions and those whose documentation either does not document all the arguments or has documentation for arguments that do not belong to the current function. This was achieved by creating two `Python` functions inside a `Python` environment called `arg_names` and `arg_doc_problems`. The first one just returns a list with the argument names from the dictionary than contains the argument names as keys and their type as value. The second one takes the dictionary of arguments and the comment with the documentation, extracts all the `@param` keywords therein and checks if both sets are equal.

The output can then be passed to a system that will take its output and generate another format. In this case, we used `pandoc` that can generate many different formats including `HTML` and `LATEX`. The following statement illustrates how to generate a report in `HTML`.

```
teaching_assistant -t report.txt PL[1-9]G[0-9]* | pandoc -o report.html
```

6 Conclusions

This tool helps instructors create reports concerning students' code. Instructors can easily create reports about all projects with this tool since it enables a lot of sophistication. With a little thought, it is capable of both grading students or provide reports to help instructors understand the problems with the students' code and better help them address it. Since most of the keywords accept `Python` code, it is very easy to extend to one's needs. As the `tabulate` model can generate tables in many formats including most markdowns, `HTML` and `LATEX`, the tables produced can be easily incorporated in many types of reports.

The main advantages of this tool is its simple syntax, it can use arbitrary `Python` code and automatically substitutes the values of fields, it can use all the statistics gathered by `Multimetric`. Tables may be generated with colors in a very straightforward fashion, they

can be generated in many formats and more advanced queries using aggregation can be used. The fact that it is possible to use markdown to create the report and have templating possibilities is also quite useful since it is possible to create small functions inside the text file for addressing a situation and embed tables inside the report.

Its main disadvantages is that it depends on other tools, like `Multimetric` and `Universal C tags`, it is currently not very graceful when encountering parsing errors and only works in Linux. It is also not capable of creating automatic advice. Future work will focus on some of the disadvantages mentioned above and to extend this tool to the Python language.

References

- 1 Sergey Astanin. Tabulate. <https://pypi.org/project/tabulate/>. Accessed: 2021-04-09.
- 2 Rachel Cardell-Oliver. How can software metrics help novice programmers? In *Proceedings of the Thirteenth Australasian Computing Education Conference-Volume 114*, pages 55–62, 2011.
- 3 Don Coleman, Dan Ash, Bruce Lowther, and Paul Oman. Using metrics to evaluate software system maintainability. *Computer*, 27(8):44–49, 1994.
- 4 Francisco Jurado, Miguel A Redondo, and Manuel Ortega. Using fuzzy logic applied to software metrics and test cases to assess programming assignments and give advice. *Journal of Network and Computer Applications*, 35(2):695–712, 2012.
- 5 Nadia Kasto and Jacqueline Whalley. Measuring the difficulty of code comprehension tasks using software metrics. In *Proceedings of the Fifteenth Australasian Computing Education Conference-Volume 136*, pages 59–65, 2013.
- 6 Xavier Marcelet. Coverxygen. <https://pypi.org/project/coverxygen/>. Accessed: 2021-04-09.
- 7 Thomas J McCabe. A complexity measure. *IEEE Transactions on software Engineering*, SE-2(4):308–320, 1976. doi:10.1109/TSE.1976.233837.
- 8 Simon, Andrew Luxton-Reilly, Vangel V. Ajanovski, Eric Fouh, Christabel Gonsalvez, Juho Leinonen, Jack Parkinson, Matthew Poole, and Neena Thota. Pass rates in introductory programming and in other stem disciplines. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education, ITiCSE-WGR '19*, page 53–71, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3344429.3372502.
- 9 Errol Thompson, Andrew Luxton-Reilly, Jacqueline L Whalley, Minjie Hu, and Phil Robbins. Bloom's taxonomy for cs assessment. In *Proceedings of the tenth conference on Australasian computing education-Volume 78*, pages 155–161, 2008.
- 10 Szymon Wasik, Maciej Antczak, Jan Badura, Artur Laskowski, and Tomasz Sternal. A survey on online judge systems and their applications. *ACM Computing Surveys (CSUR)*, 51(1):1–34, 2018.
- 11 Konrad Weihmann. Multimetric. <https://pypi.org/project/multimetric/>. Accessed: 2021-04-09.

A List of keywords

This appendix shows a list of the keywords that can be used by the tool. The software engineering statistics that are provided by `Multimetric` include measures that are only available for other languages and thus are omitted from this list. The keywords are the following:

name Function name

folder The path of the project containing the file where the function was defined

project The basename of **folder**

filename The path of the file where the function was defined

filetype Whether the function was defined in a C file or include file

13:8 A Teaching Assistant for the C Language

return Function return type
args Dictionary of function arguments and their types
comment Comment before the function that may be the Doxygen documentation
vars Dictionary of local variables and their types
stat Software engineering statistics provided by **Multimetric** for each function
 comment_ratio Percentage of comments
 cyclomatic_complexity Cyclomatic complexity according to McCabe
 halstead_bugprop Number of delivered bugs according to Halstead
 halstead_difficulty Difficulty according to Halstead
 halstead_effort Effort according to Halstead
 halstead_timerequired Time required to program according to Halstead
 halstead_volume Volume according to Halstead
lang Programming language
loc Lines of code
maintainability_index Maintainability index
operands_sum Number of used operands
operands_uniq Number of unique used operands
operators_sum Number of used operators
operators_uniq Number of unique used operators

SHREWS: A Game with Augmented Reality for Training Computational Thinking

Francisco Saraiva ✉ 

Centro ALGORITMI, Departamento de Informática,
University of Minho, Campus Gualtar, Braga, Portugal

Lázaro V. O. Lima ✉ 

Centro ALGORITMI, Departamento de Informática,
University of Minho, Campus Gualtar, Braga, Portugal

Cristiana Araújo ✉  

Centro ALGORITMI, Departamento de Informática,
University of Minho, Campus Gualtar, Braga, Portugal

Luis Gonzaga Magalhães ✉ 

Centro ALGORITMI, Departamento de Informática,
University of Minho, Campus Gualtar, Braga, Portugal

Pedro Rangel Henriques ✉  

Centro ALGORITMI, Departamento de Informática,
University of Minho, Campus Gualtar, Braga, Portugal

Abstract

This paper proposes a game to help young students training Computational Thinking (CT) skills to aid in solving problems. CT is a problem-solving approach based on picking a complex problem, understand what the problem is, and develop solutions in a way that a computer or human could solve. To help in this task, Augmented Reality (AR) will provide a more engaging visual way of interaction to keep students motivated while they search for solving problems. This benefit is a consequence of the AR capability of providing a visual and dynamic representation of abstract concepts. This work investigates AR and CT concepts and the best way of combining them for training student's skills to understand software and its effects. Thus, these concepts will be explored for the construction of learning activities to explain and create analogies to understand complex concepts related to computer programs. So the focus of the paper is the introduction of Shrews, the game created in this context. The principle and the proposed architecture are detailed. At the end, there is a description of how the game works and the current state of the prototype. We believe that the immersive experience using AR and CT concepts is one of the important aspects of the game to maintain a motivational approach to students. An exploratory prototype is created to explore the topic of teaching CT skills via playing a video game.

2012 ACM Subject Classification Computing methodologies → Mixed / augmented reality; Social and professional topics → Computational thinking

Keywords and phrases Augmented Reality, Computational Thinking, Learning Activity, Game

Digital Object Identifier 10.4230/OASICS.ICPEC.2021.14

Category Short Paper

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

Computational Thinking, is the action of “thinking like a computer” [15], but in detail refers to the process of solving problems following taking knowledge and computational practices such as, systematization, representation, analysis, abstraction, pattern recognition, solving and interpreting problems.



© Francisco Saraiva, Lázaro V. O. Lima, Cristiana Araújo, Luis Gonzaga Magalhães, and Pedro Rangel Henriques;

licensed under Creative Commons License CC-BY 4.0

Second International Computer Programming Education Conference (ICPEC 2021).

Editors: Pedro Rangel Henriques, Filipe Portela, Ricardo Queirós, and Alberto Simões; Article No. 14; pp. 14:1–14:10



OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

On the other hand, and since its inception Augmented Reality(AR) showed promising in its use, but has not seen much popularity to its existence to the every day person. According to even by the very early uses back in [1], over the years AR has seen mild or small use but steady increase in its popularity and uses. Azuma defined AR by three characteristics: **Combines real and virtual, Interacts in real time and is supported in 3-D**. AR [3, 1], is a variation of Virtual Reality in which it allows the user to see the real world but with virtual objects imposed onto the real world as to integrate the digital into the real.

So, the objectives of the project here reported are study AR and the impact of building games to train Computational Thinking (CT); Develop a game based in AR and conduct tests to study the benefits of CT. The project outcome is expected to be a Learning Activity that can where there is a game that will help to develop the four characteristics of the CT described above. To guide the research, DSR (Design Science Research)[13] methodology was used.

The phases of the work are based on the DSR methodology and guidelines of the artefact design, understanding of the problem, evaluation of the product and the contribution. The project is split into a study of the technologies and topics of CT and AR, followed by the conception of the product to solve the question of the possibility of training CT via a video game and the analysis and testing of the product with subjects.

The paper is organized as follows: in Section 2 the importance of Computational Thinking to improve the humans' ability to solve problems using the computer is discussed; in Section 3 we describe the research work already done combining Augmented Reality and CT; in Sections 4 and 5 we introduce and describe the proposed game, Shrews, explaining how does it work, as well as the concepts of CT that it addresses; and in Section 6, some sketches illustrating how the game can be played are shown.

2 The importance of Training Computational Thinking with the appropriate resource

Pragmatics shows that a person only acquires a new way of thinking or behaving if he is trained with the appropriate devices, for that, we will use the concepts of Computational Thinking to build a game appropriate for training CT skills; The concepts are: *Pattern Recognition* is the procedure of recognizing common characteristics or traits in problems and solutions. This leads to finding possible and efficient solutions to problems. *Decomposition* is the process of splitting a bigger problem into smaller ones, which can then be solved individually and possibly solve the bigger problem together. *Algorithms* are sets of instructions for a determined solution of a problem. Lastly *Abstraction* is the ability of separating elements in a problem as a way of organizing the data structures to better understand the complexity and model of the problem.

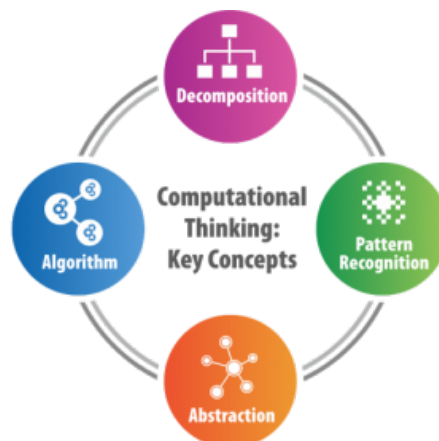
CT has found its way into a lot of research agenda of all science and engineering disciplines. The author also say that CT as found it is way into a lot of other important professions and areas such as medicine, economics, finance, law, arts and even digital humanities and digital journalism. Data mining techniques are also heavily based on the CT process and its four principles. For that reason and the growing abundance of data to be processed in current time [7], due to the accessibility of computers and internet connections, developing these skills is important for future graduates and students alike to tackle these activities in their future jobs and professions. After showing the concepts about CT, in the next section, we present which characteristics of these related concepts can be explored using AR in Shrews.

3 Augmented Reality and Computational Thinking

Virtual Reality and Augmented Reality make up the Computerized Hybrid Worlds, these technologies fascinate and are absorbed naturally by society in the most varied ways and integrate into our everyday life, taking over new technologies to perform human tasks. AR is defined by Azuma [1] as the overlapping of virtual information in the real world through technology; and [4] shows the directions of future works and areas that are currently being researched. This information can be simple textual images or 3-D objects.

The author had seen numerous potential in it's uses in technology be it medical, manufacturing, repairs, entertainment, planning and military, which today, years later, does see much more use in our society [2, 10] such as in DIY car repairs, cooking aids, GPS navigation, video games, medical training, car mechanics, customer service, military airplanes, television broadcasts and even house interior modeling.

Augmented Reality has the characteristic of creating attractive interactive interfaces that we can interact without using conventional peripherals and thus providing greater interaction between users and impacting their motivation. AR supports pedagogical approaches through Constructivist learning by enabling educational experiments that complement the activities of the real classroom. Thus it is possible to explore AR as a technology that provides a wide range of skills such as problem-solving, abstraction, creative manipulation of the computer as a user, not only as a mere technology operator but as a computationally literate individual. In Figure 1, we see the main concepts of CT.



■ **Figure 1** The four concepts of Computational Thinking.

In the research of results by using Video Games as tools, the study by [12] on the hypothesis if playing puzzle Video Games could improve executive functions, results showed that playing a complex puzzle game that demands strategy, problem interpretation and analysis improved several aspects of executive function. It is assumed that due to the variations of difficulty, strategies and problem solving methods may have led to a greater learning and generalization of the tasks used to test changes in the cognitive and control networks in the brain, responsible for high problem solving and executive functioning. Another study, this time more directed at fostering CT skills in middle school students by [16], it was discovered that by playing Penguin Go for less than two hours a day, improved students CT skills significantly, yet results on the impact of learning were inconclusive, but it might support the study on academic success and CT by [8] as being a factor of culture and the differences in the grading system on how we measure student success.

As AR has a steady increase in popularity and the use of this technology becomes more prominent in our lives, the question of its use in helping with education also comes into play. We know of its use in training and educating medical professionals but other areas or varying degrees and levels of education can see the potential and use of the technology. On work of [5] it's discussed how AR is used in learning and pedagogical approaches and exercises with lower thinking order as for example remembering, understanding and applying while high thinking orders of analysis, evaluating and creating have not been so thoroughly explored. Such activities include constructive learning, situated learning, games based learning and inquiry-based learning.

Games like CodeCubes [6] and HyperCubes [9] are both two game based solutions that employ AR to effectively train and teach programming and computational concepts respectively. CodeCubes is a game with an interface to stimulate CT in its users utilizing AR markers to program three different levels. HyperCubes is also an AR centered game that focuses on helping children understand computational concepts by interacting with their physical surroundings by utilizing the various commands in game and real cubes with AR markers. It plays more akin to a sand box like game to experiment. Both are game that utilize AR to teach and train these concepts and have had some success proving the use of these technologies in a learning environment.

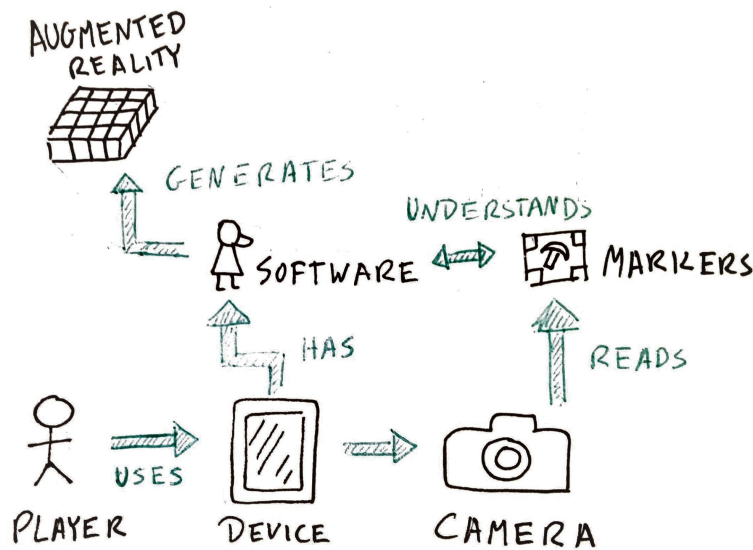
A research on the use of AR in more commercial Video Games was done to better understand what constitutes a good use of AR and its interaction. A number of popular AR Video Games on the market were picked and studied in their uses of this technology in their game play. Some Video Games were found to use AR in a “good” and “bad” way, with good being the use of AR according to its definition of integrating the virtual with the real and the augmentation of the interaction in the Video Game such as using perspective of the camera to analyze the virtual object to solve a problem, while bad being a minimal use of the the technology as simple as super imposing the 3-D but not offering any other interaction other than viewing pleasure. Based on the research and our analysis of what constitutes good use of AR in a Video Game, the genre decided upon was to be a 3-D puzzle, that will use perspective and AR markers in its core game play to keep the engagement and interaction with the technology, which will be further explored in the following section.

4 Shrews: Game with AR for Training CT

After researching Video Games centered around AR, this section will propose the artifact intended to be developed based on the gathered information, data and research to develop a Video Game that will aim to train and focus on the four principles of and AR centered functionalities and engagement.

To develop a proposal of a Video Game to train CT, we have to look at the four main characteristics that compose Computational Thinking: **decomposition, abstraction, algorithms and pattern recognition**. In Video Games that tackle these four characteristics there is a genre that stands out among them and it is puzzle games, which make the player think and understand various concepts in the game to solve problems with the tools available. A puzzle Video Game that tackles these concepts is Lemmings, a 2-D puzzle-platform Video Game developed by DMA Design in 1991, in which the objective is to guide a group of characters through various obstacles in a 2-D view from a point A to a point B.

In a research on Scalable Game Design (SGD) used to give young students CT acquisitions by [11], Lemmings was used as a tool to train problem solving skills on some success for young students for their better understanding of computational skills and decision making.

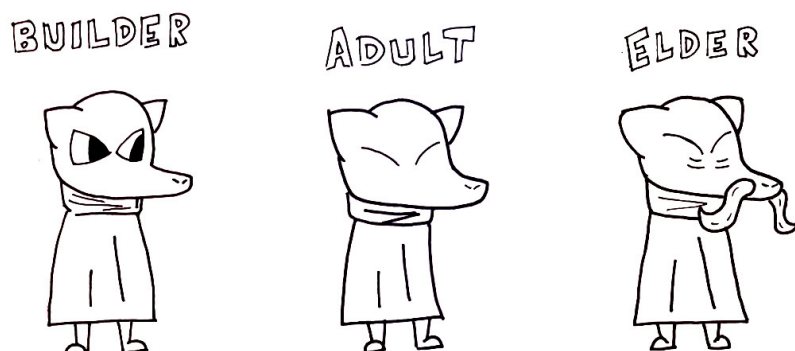


■ **Figure 2** The Scheme of Shrews and the various components that make up it.

The Figure 2 represents the schema of the game, by components and interaction. The components that make up this system are the **device**, such as a smartphone or tablet, in which the player will use with the **software** that will be developed. This software will be tasked with the execution of the Video Game and creation of the AR environment that the player will use as a lens to see it. The game will use a camera to track a surface to project the AR environment and identify the various AR markers to execute the Shrew's tasks.

5 Game - How will the game work?

The Shrews are three types of characters: **Builder**, **Adult**, and **Elder** with different skills see in Figure 3.

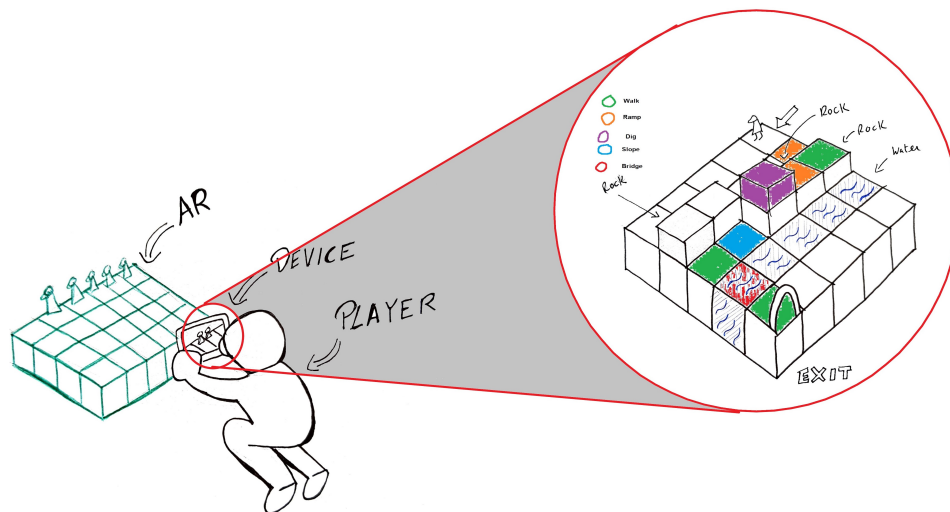


■ **Figure 3** Three types of characters of SHREWS Game.

The objective of the game is to transverse a group of Shrews over the map, from the start to the exit. For this the player will use AR cards to instruct the **Builder** where to build such pathways for the Shrews. The **Adult** and **Elder** Shrews are moved by how the **Builder**

guides them safely across the map, as they cannot see in front of them. The player will find a flat surface preferably, to generate a map and will use perspective to analyze the map and terrain and figure out the best paths around the map to guide the Shrews. The player will focus on the Builder or the blocks and use cards to instruct which action the Builder will do. For example tapping on a block and choosing a card will make the builder move to that spot to build the path forward. The remaining Shrews will walk back and forth nonstop on the pathway the Builder shrew has moved in. The game will have a scoring system based on various statistics throughout the session like time it took to solve the puzzle, how many tasks were used and how many Shrews made it safely to the end.

The Figure 4 shows how the player will use a device to see the AR. The player will use a smartphone device or tablet as a scope into the game and see the AR map and game play happening. Moving with the device around the map the player will be able to see the whole map from different perspectives. Using the AR cards to read from the application will cause the various tasks of the builder shrew to be executed.



■ **Figure 4** Example of user interaction with the game.

The game will be built with more than one way to solve the same problem. The map holds various normal dirt blocks, rock blocks and rivers or ponds composed of water blocks. We will go over many scenarios to solve this puzzle, but there can be more than one solution to a puzzle, it is up to the player to find them to promote each interpretation of the problem.

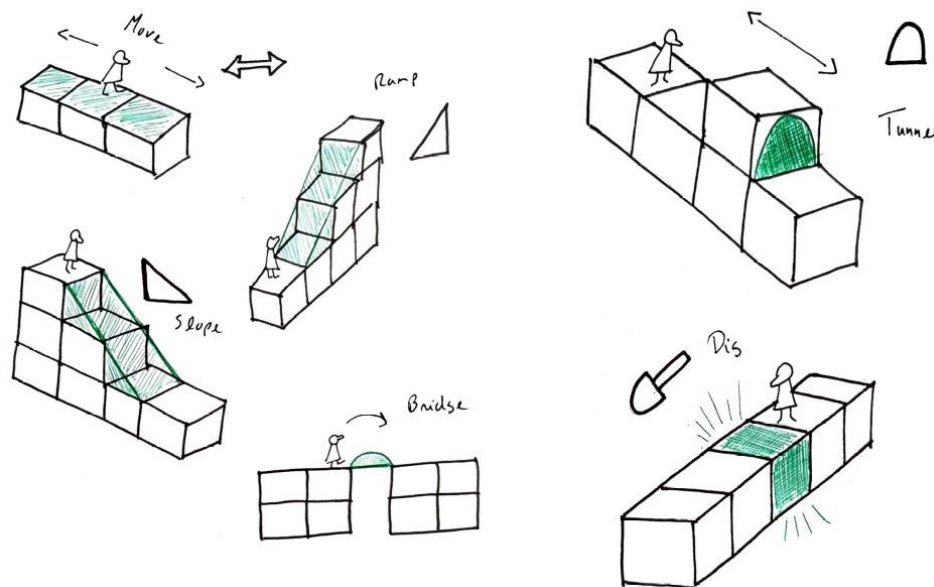
The skills the Builder can perform, illustrated in figure 5 are as follows:

- **Move:** Instructs which blocks the shrews can navigate, the player will tap the blocks and use the AR card to signal these blocks are for the builder to instruct the shrews to walk on.
- **Ramp:** Creates next to the selected, or more, dirt blocks a ramp allowing the shrews to go up or down in height to another block, depending on the direction they are walking from they will climb, or slide down the ramp. It is to note that the builder shrew does not slide down the ramps, only the adults and elders, depending on how many ramps elders slide down the more hurt they get on landing. These are the only buildings the builder Shrew can construct without needing to shape a block.
- **Bridge:** Constructs a bridge over blocks of water or more connecting blocks over a space, allowing the shrews to walk over the water blocks.

- **Tunnel:** Builds a tunnel through a block to move to the other side, this is only possible on dirt blocks and not rock blocks.
- **Dig:** Destroys a selected block near the builder shrew to open up space on the map for construction.

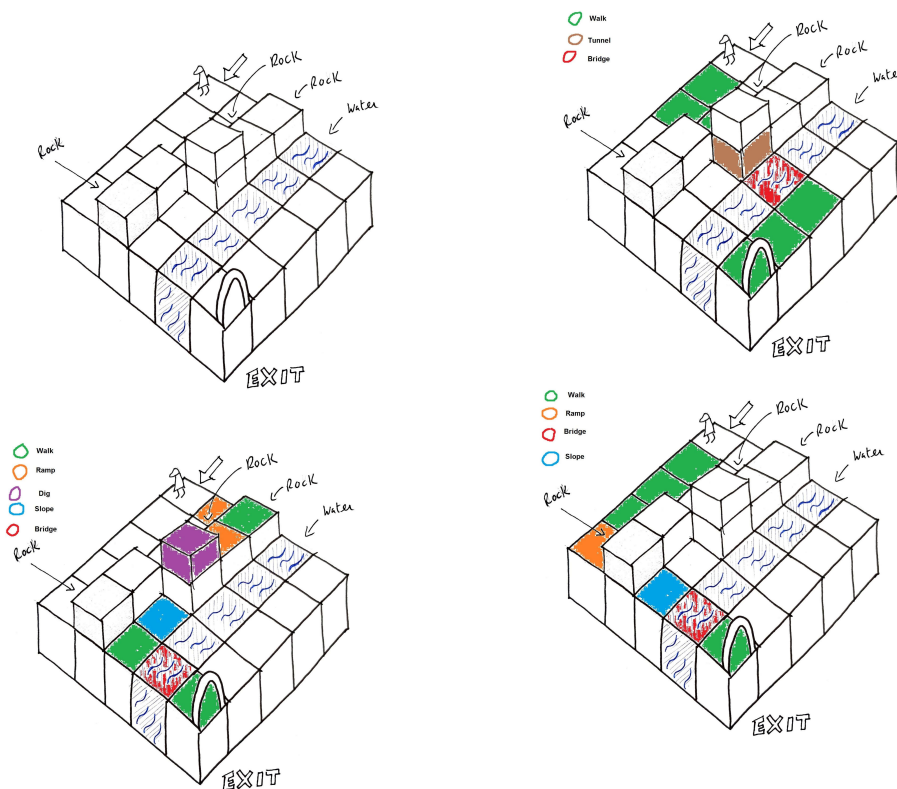
As the objective is to train CT skills in the players, the game play will focus on the four thought processes, this way the players will experience the problem solving methods used in CT and train them in game to solve puzzles.

- **Pattern Recognition:** By utilizing skills and how they interact with the blocks to give pathways, players will start recognizing patterns. Observing the behavior of the shrews as they are patterned will help to guide them better.
- **Decomposition:** The puzzle is a large problem with smaller problems in it via obstacles such as blocks in higher heights need to be climbed or other obstacles that require solving.
- **Algorithm:** The objective of the game is to trace an algorithm that will lead the Shrews from point A to point B.
- **Abstraction:** The difficulty of the puzzles are measured by the number of existing blocks and overall dimension of the map. It's important to abstract how flat surfaces of blocks can be paths as opposed to groups of blocks in different heights, as well as the different types of terrain. Also the perspective view of AR to visualize the map in all angles gives the player the sense of analysis and strategic planning.



■ **Figure 5** The first and second set of skills the Builder can perform.

In this example, Figure 6, it has been illustrated in colors the blocks and the tasks associated that the builder shrew will perform. Illustrated in green are the blocks indicated to walk over, followed by a block in brown that the builder will construct a tunnel, followed by red, a bridge to cross the water and the remaining tiles in green is to walk over to the exit. This is a more simple approach to this puzzle as it is straight forward, without going up in height and down to cross the the higher blocks.



■ **Figure 6** Example of a types of maps and solutions to the puzzle.

6 Prototype in Progress

As to better classify and organize the proposed game an ontology for game classification was used called “**Ontojogo**” by Teixeira [14], as a system to better classify the properties and attributes of games. Using the expertise already investigated by the authors, we can detail the characteristics of the game through this Ontology that classifies the games according to different perspectives or different categorization axes.

■ **Listing 1** Classification of Shrews in **OntoJogo** by [14].

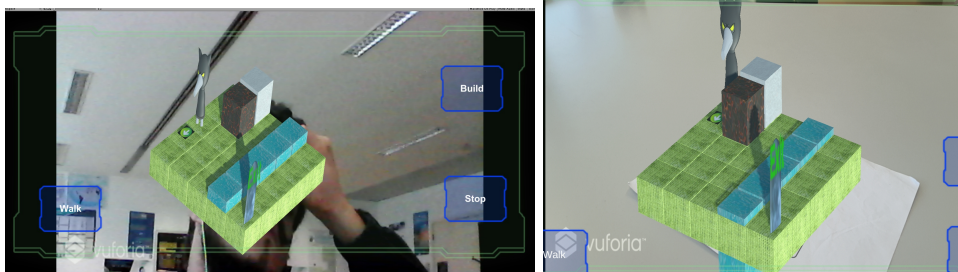
```

Shrews =iof => Game[name='Shrews', description='Shrews is a puzzle
solving game where the player helps guide creatures over a map.',
age_rating=7, storyline='fiction'];
Shrews =is_available => Digital Offline;
Shrews =has_mode => Casual;
Shrews =has_perspective => Third-Person;
Shrews =with_input => Movement-Based;
Shrews =with_input => Touch;
Shrews =has_player_number => Single Player;
Shrews =has_progression => Level-Based;
Shrews =belongs_to_platform => Mobile;
Shrews =belongs_to_genre => Puzzle;
Shrews =belongs_to_genre => Strategy;

```

We believe that the game, already classified within the ontology to guide the Computational Thinking approach, will be motivating and a tool with a positive impact. To implement the ideas presented here, we use Unity 3-D® v.2018.4, the tool builds Augmented Reality

systems supported by the Vuforia® version 9.8 library. We can see in Figure 7, the conceptual prototype of the game. In this scenario we have the shrews in a terrain with obstacles to be overcome and the options for the skills that the Builder Shrew can perform on the map.



■ **Figure 7** Prototype of the game currently in progress.

After the functional prototype is ready to be playable, it will be used to test on a demographic of young students to explore their interactions with the game and how they interpret the game and functionalities as to further improve the product to reach our results.

7 Conclusion

Over the course of this paper CT and related topics its definitions have been discussed emphasizing the four main concepts Algorithm, Decomposition, Pattern Recognition and Abstraction, its growing importance and interest of study in our modern society as well as the benefits to develop these skills. The it was discussed how Video Games help shape our way of thinking improving different and relevant strategies for with their intricate game play and thought process to solving problems. Previous games, like CodeCubes and HyperCubes, have proven to be successful in teaching CT due to the AR models they use; however that process was experimented with children working under specific technological environments like computer programming clubs. On the other hand, Shrews aims at making the difference with a real personality, able to captivate a wider range of people; to attain that purpose , Shrews has a proper design and style,as well as easy to use interface.

Along the paper it was also discussed AR, its variants and its uses. The growing popularity of the technology in our society was pointed out, as well as the Video Games that utilize it in their core and employ good uses of the technology in their interactions and practice. Combining the two subjects the our proposal, here presented, is to develop an AR 3-D game that will train its players to develop CT skills by adding the four main principles into the game. With a simple prototype being currently developed we hope for our next iteration to be a playable version of our vision for this Video Game to help train and develop the Computational Thinking skills of individuals with a immersive and interesting technological approach to hold and promote interest.

References

- 1 Ronald T Azuma. A survey of augmented reality. *Presence: Teleoperators & Virtual Environments*, 6(4):355–385, 1997.
- 2 Jacob Bell. 4 real world uses for augmented reality - primacy primacy | blog. <https://www.theprimacy.com/blog/finally-4-real-world-uses-for-augmented-reality/>. (Accessed on 04/06/2021).

- 3 Donna R Berryman. Augmented reality: a review. *Medical reference services quarterly*, 31(2):212–218, 2012.
- 4 Mark Billingham, Adrian Clark, and Gun Lee. A survey of augmented reality. *Foundations and Trends in Human-Computer Interaction*, 2015.
- 5 Matt Bower, Cathie Howe, Nerida McCredie, Austin Robinson, and David Grover. Augmented reality in education—cases, places and potentials. *Educational Media International*, 51(1):1–15, 2014.
- 6 Bárbara Cleto, Cristina Sylla, Luís Ferreira, and João Martinho Moura. CodeCubes: Coding with Augmented Reality. In Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões, editors, *First International Computer Programming Education Conference (ICPEC 2020)*, volume 81 of *OpenAccess Series in Informatics (OASICs)*, pages 7:1–7:9, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/OASICs.ICPEC.2020.7.
- 7 Debra J CPCM and Philip Noah. Big data in the information age: exploring the intellectual foundation of communication theory. *Information Systems Education Journal*, 12(1):15, 2014.
- 8 Hao Lei, Ming Ming Chiu, Feng Li, Xi Wang, and Ya-jing Geng. Computational thinking and academic achievement: A meta-analysis among students. *Children and Youth Services Review*, 118:105439, 2020.
- 9 Anna Fusté Lleixà. Project overview ' hypercubes: Learning computational concepts in augmented reality, 2018. URL: <https://www.media.mit.edu/projects/hypercubes/overview/>.
- 10 Christopher McFadden. 9 powerful real-world applications of augmented reality | ie. <https://interestingengineering.com/9-powerful-real-world-applications-of-augmented-reality>. (Accessed on 04/06/2021).
- 11 Hilarie Nickerson, Catharine Brand, and Alexander Repenning. Grounding computational thinking skill acquisition through contextualized instruction. In *Proceedings of the eleventh annual International Conference on International Computing Education Research*, pages 207–216, 2015.
- 12 Adam C Oei and Michael D Patterson. Playing a puzzle video game with changing requirements improves executive functions. *Computers in Human Behavior*, 37:216–228, 2014.
- 13 Rauno Pello. Design science research — a short summary | by rauno pello | medium. <https://medium.com/@pello/design-science-research-a-summary-bb538a40f669>. (Accessed on 04/07/2021).
- 14 Salete Teixeira, Raul V Boas, Francisco Oliveira, Cristiana Araújo, and Pedro R Henriques. Ontojogo: An ontology for game classification. In *2020 IEEE 8th International Conference on Serious Games and Applications for Health (SeGAH)*, pages 1–8. IEEE, 2020.
- 15 Jeannette M Wing. Computational thinking benefits society. *40th Anniversary Blog of Social Issues in Computing*, 2014:26, 2014.
- 16 Weinan Zhao and Valerie Shute. Can playing a video game foster computational thinking skills? *Computers & Education*, 141:103633, July 2019. doi:10.1016/j.compedu.2019.103633.

Understanding Effects of the Algorithm Visualized with AR Techniques

Lázaro V. O. Lima ✉ 

Centro ALGORITMI, Departamento de Informática,
University of Minho, Campus Gualtar, Braga, Portugal

Manuel Sousa ✉ 

University of Minho, Braga, Portugal

Luis Gonzaga Magalhães ✉ 

Centro ALGORITMI, University of Minho, Braga, Portugal

Pedro Rangel Henriques ✉ 

Centro ALGORITMI, Departamento de Informática,
University of Minho, Campus Gualtar, Braga, Portugal

Abstract

We create analogies to understand and visualize complex concepts. Such approach, based on analogies are presentation of software, is also effective when it concerns software comprehension. Many visualization techniques for data structures have been developed in 2D and 3D to improve the visual representation of large structures. A common challenge faced by developers that want to implement these techniques is to increase the amount of information to be displayed in each node seeking a balance between quantity and visibility. To overcome these challenges, this article presents a visualization technique using Augmented Reality to display hierarchical structures and understand the effects of the algorithm in data structures. The visualization system based on AR, proposed and discussed along the paper, allows the user to interact and navigate through the structure, enabling him to explore information in depth.

2012 ACM Subject Classification Computing methodologies → Mixed / augmented reality

Keywords and phrases Augmented Reality, Learning Resource, Data Visualization, Syntax Tree Visualization

Digital Object Identifier 10.4230/OASICS.ICPEC.2021.15

Category Short Paper

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 start, we quote the authors [5] who say “The use of computer-supported, interactive, visual representations of abstract data to simplify cognition” – in other words, we create analogies to understand complex concepts and this relationship is a cognitive process of information transfer.

For this, another author [4] says: “Visualization holds great promise for computational science and engineering, provided we can meet the immediate and long-term needs of both toolmaker and tool users.” According to [7], our cognitive system takes about 15 to 20 different psychological stimuli into account to perceive the spatial relationships between 3D objects. These so-called depth tips can be divided into monocular (use of one eye) and binocular (use of both eyes). This information is usually classified into 3 categories: one-dimensional, two-dimensional and three-dimensional [6].



© Lázaro V. O. Lima, Manuel Sousa, Luis Gonzaga Magalhães, and Pedro Rangel Henriques;
licensed under Creative Commons License CC-BY 4.0

Second International Computer Programming Education Conference (ICPEC 2021).

Editors: Pedro Rangel Henriques, Filipe Portela, Ricardo Queirós, and Alberto Simões; Article No. 15;
pp. 15:1–15:10

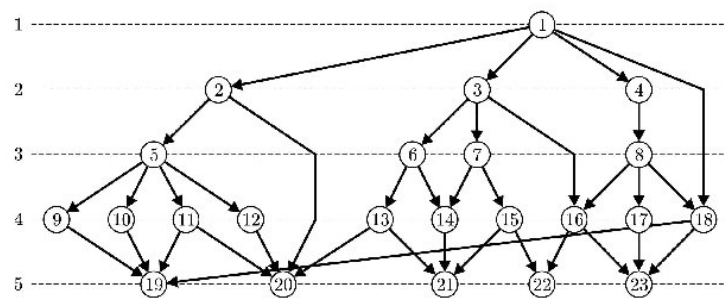


OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

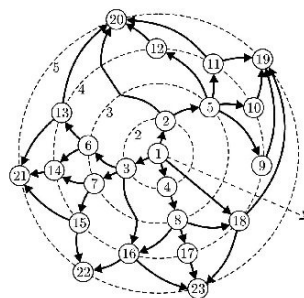
The form of representation of certain information must be carefully chosen since misrepresentations can cause errors of interpretation [14]. Based on these categories, several metaphors were created, such as bar graphs, hierarchical structures (*Information Cube* [11], indented lists [13], hierarchical graphs [3], *TreeMaps* [8], *ConeTree* [1]) and maps to represent the different categories of information.

Indented lists, the items (nodes) belonging to the hierarchy are organized one below the other, linearly, with the indentation corresponding to the level occupied by the element in the hierarchy [13]. A hierarchical graph can be defined as a graph where each node is composed by a simple element or, in turn, a new hierarchical graph [10]. When a graph has the purpose of visualizing hierarchical data, it is called a hierarchical graph, and can be presented in different ways (horizontal and radial). The horizontal hierarchical graph, as shown in Figure 1, has the level lines drawn horizontally, parallel to each other.



■ **Figure 1** Horizontal hierarchical graph [3].

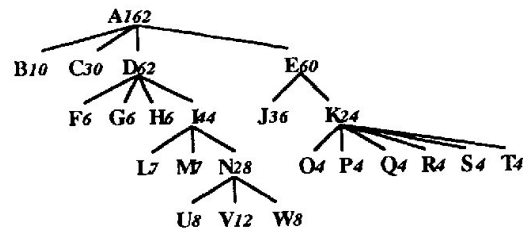
To solve this problem, the radial hierarchical graph presented in Figure 2 was proposed and created by Bachmaier [3], having the level lines represented by concentric circles. One disadvantage of radial hierarchical graphs is that for large volumes of nodes, clarity in the visualization of the information presented is lost.



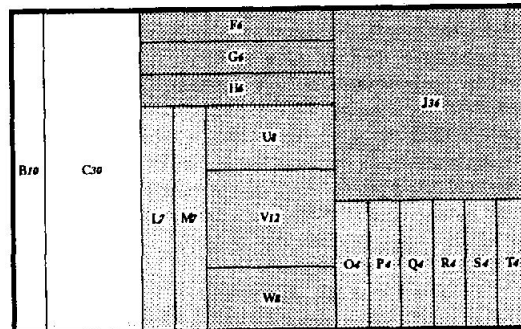
■ **Figure 2** Radial hierarchical graph [3].

The *TreeMaps* presented by Figures 3 and 4, constitute a structured information visualization technique, which maps a given hierarchy in a rectangular area, using 100% of the available space for the presentation of data [8]. With the efficient use of available space, it becomes possible to view large hierarchies, facilitating the presentation of the information in question.

Even though this technique is a good option for viewing large hierarchies, it becomes limited when displaying a single type of information throughout the structure (viewing in two dimensions limits the ability to present complementary information).

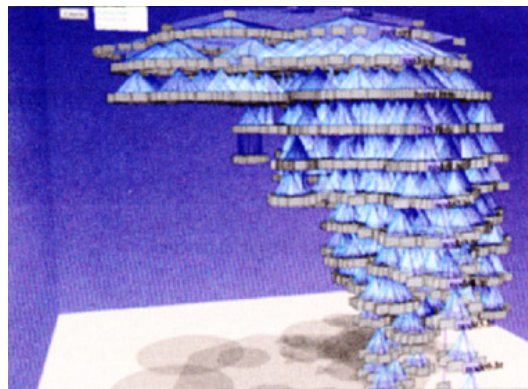


■ **Figure 3** *TreeMap* equivalent to the structure of Figure 4 [8].



■ **Figure 4** Example of *TreeMap* [8].

ConeTrees are a technique for visualizing hierarchical structures, through the connection of sub-trees, in the form of 3D cones, as shown in Figure 5. The main disadvantage of *ConeTrees* is the occlusion of some nodes, which increases significantly as larger hierarchies are viewed. To work around this problem, *ConeTrees* are designed to be interactive, allowing the user to navigate through the presented structure and, consequently, the visualization of occluded nodes [1].



■ **Figure 5** Example of *ConeTree* [1].

The techniques described above have some limitations, such as:

- Presentation of a single type of information throughout the structure;
- High degree of disorder in the presentation of different types of information;
- Does not allow interaction with the hierarchical structure presented.

Therefore, we need to present new techniques for visualizing information with Augmented Reality. We understand that the AR allows the observation of a data structure in three dimensions, in addition to allowing the observation of changes that occur in the data structure knowing that the AR inserts increased information in real time. Augmented Reality is defined by Azuma [2] as the overlapping of virtual information in the real world through technology. This information can be simple textual images or 3D objects. 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.

The prototype proposed in this article addresses a different use of the visualization technique of data structures, in which the user interacts with the information presented at different levels of depth through a Virtual Environment. The goal is to allow the representation of arbitrary information and data.

Taking advantage of the best practices of Augmented Reality, one can align the objectives of teaching Algorithms and their different data structures with 3D visualization in which the teacher can perform an operation and instantly visualize the structure in Augmented Reality. Besides this introduction, this paper is divided into five sections: the section 1 covering the importance of using technologies that impact students' motivation to understand complex concepts and thereby make the abstract better; another one documenting the construction and presentation of tree information; a third one presenting the application of the prototype built with a case study and a last one to show some concept prototypes, conclusions and work in progress.

2 Understanding Software with Visualization of Effects in AR

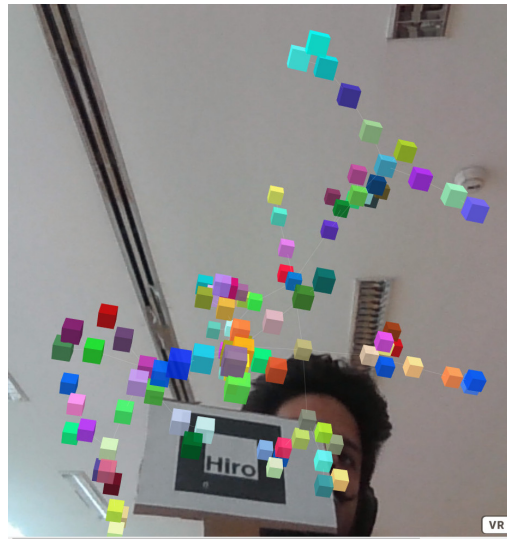
There is a lot of research underway on 3D data visualization in AR, especially in the field of Immersive Analysis and analysis of medical volume, such as Sielhorst [12]. Luboschik [9] claims that among the fundamental aspects to be considered in Augmented Reality are linear perspective, relative sizes, parallax of movement, binocular disparity. It is also important to show here that AR allows us to understand how a program works, that is, the effect of an algorithm on a data structure. To illustrate the visualization technique presented in this work, the visualization of family trees in 3D was chosen as a case study. In times of pandemic, a tool that shows medical information can prove to be useful for relating, for example, hereditary and relational diseases.

The purpose of the proposed visualization technique is to allow the user to understand the effects of an algorithm on a visualization structure and / or database, reducing the degree of visual disturbance. Unlike a two-dimensional presentation, in which a greater amount of information, colors and shapes tend to negatively influence the abstraction of information by the user, the three-dimensional system allow a amount of data, colors and shapes, without leaving the environment overloaded with secondary information.

Once the developed prototype is executed from the *browser*, it is possible to add external content to the hierarchy, presented as web links, images, complementary texts, among others. We will present in the next topics the tools used to validate the prototype proposed here.

3 Construction and presentation of the tree

With the prototype, the user can build a tree containing different types of information. The library *AR.js* is a lightweight open-source library for Augmented Reality on the Web, which supports features such as Image Tracking. It is essentially a JavaScript framework acting as a port of ARToolkit. Their goal is to be able to use AR on web browsers without losing

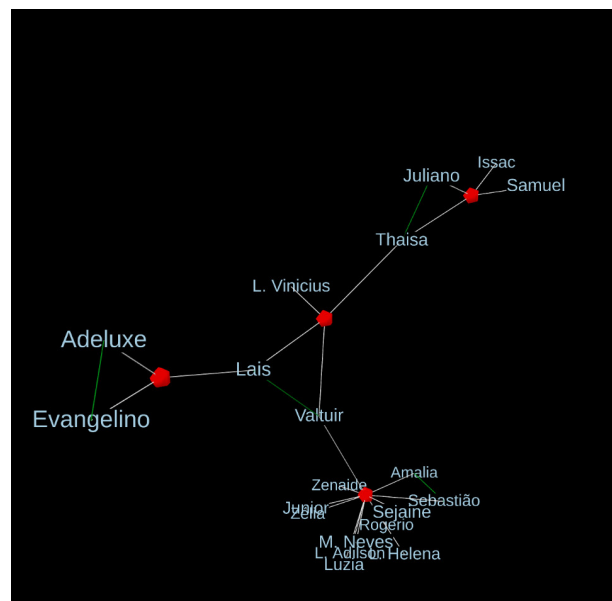


■ **Figure 6** Example of Graph with links in Augmented Reality.

performance. One of the advantages of using this type of technology is that it can run on old hardware, such as old smartphones, enhancing user accessibility. The library is web-based, using components such as **Three.js** + **A-Frame** + **jsartoolkit5**.

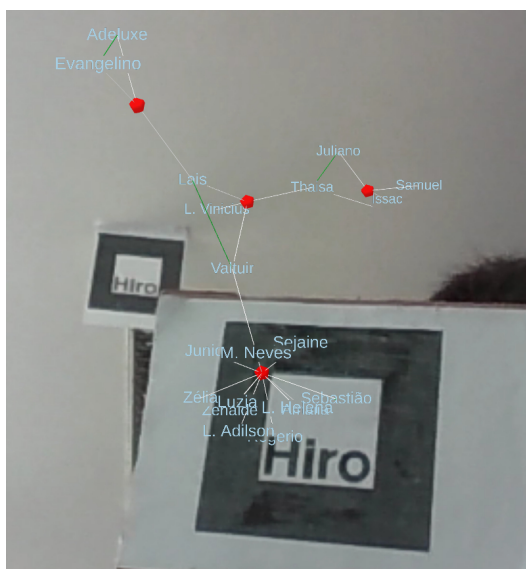
The prototype is using 3D Force-Directed Graph in AR, which is a web component to represent a graph data structure in Augmented Reality using a force-directed iterative layout. This makes use of *AR.js* with A-Frame for rendering and *d3-force-3d* for the layout physics engine.

Each node in the hierarchy can represent any type of data. This information is distributed in volumes. Moreover, in the case of two nodes being more closely connected, we created a type of relation named “marriage”, which also has the literal meaning in a use case of a family tree. This relation is represented by a node with a color red as seen in Figure 7.



■ **Figure 7** Example of Family tree in Virtual Reality.

After the tree is generated, the user can interact with the nodes and said information by accessing different context menus. Within these menus, it is possible to search the information of each node as well as insert a new node. The system is also capable of saving or loading a tree in a specific *JSON* format see in Figure 8.



■ **Figure 8** Example of Family tree with Augmented Reality.

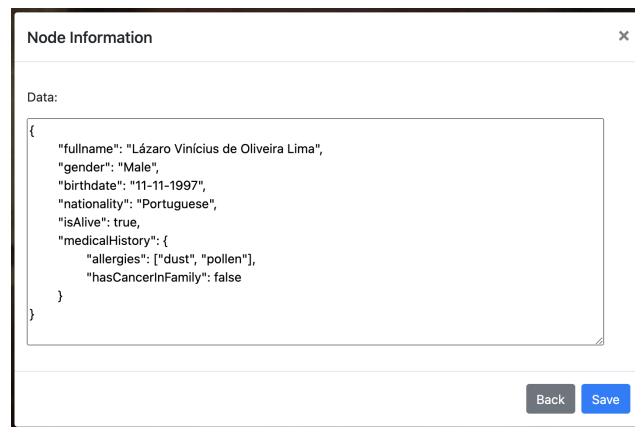
It is important to emphasize that the prototype allows the addition of a new element, the visualization of the addition of that element is animated and happens in real time. Taking advantage of the best practices of AR, it turns out that the effects can be used to teach structures such as Lists, Stacks and Queues. This concepts are better understood by students if we use appropriate analogies created by AR.

An important attribute of virtual systems is their navigation and interaction with the user. Once the family tree is built, the user can access, within each node, the complementary information, for example, personal data, professional data, or medical data. Each node has a *JSON* object representing the information that it contains. To view and edit the node information, a text area is available for the user, as seen in Figure 9. After saving new data, it is verified if the input of the user has a correct *JSON* format.

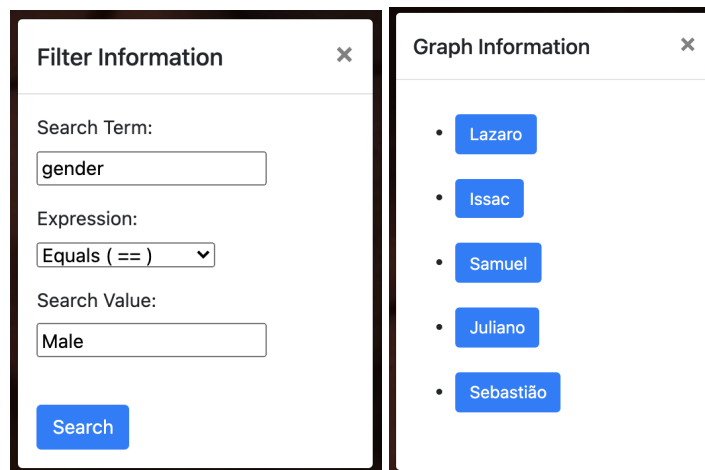
Using the tool, detach the nuances of using Augmented Reality to the usual one used to increase and decrease the scale of the graph. Navigation allows the user to identify connection points, thus locating family members. The Figure 9 presents one of the chosen family nodes and the personal data area. When the hierarchical structure has a large number of nodes, and even with the existing interaction resources inherited from the applied AR techniques, it was created an easy and understandable way for the user to visualize a specific node – a filter (Figure 10) using a search term and evaluating said term with a value.

Using a search term that corresponds to a key in the information object, an expression that can evaluate the filter as “Equals”, “Not Equals”, “Greater than” and “Less than”, and finally a search value, the user can search for specific nodes that match the previous created filter.

After presenting the example described for the construction of the visualization of connected structures and with the possibility of adding information on each element, in the next session, we will present the representation of a tree of visual syntax analysis in AR.



■ **Figure 9** Visualization of one of the nodes in the tree and personal data form.

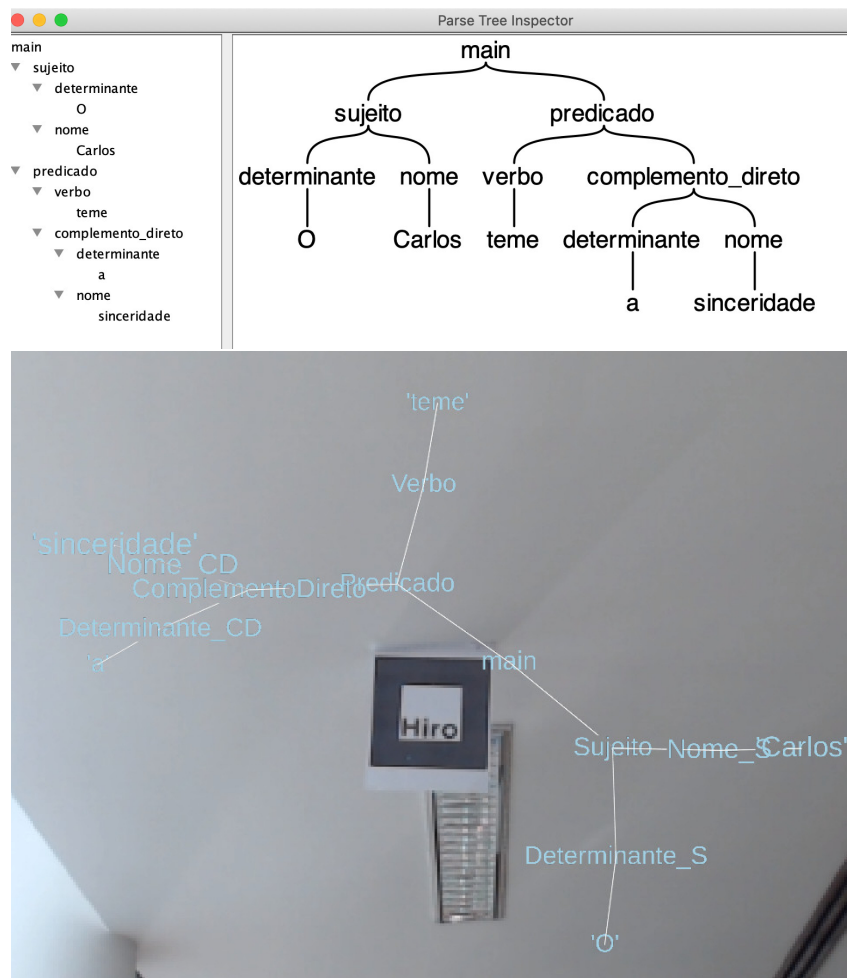


■ **Figure 10** Nodes information of Nodes and Filter Information Panel.

4 Lyntax – Use Case

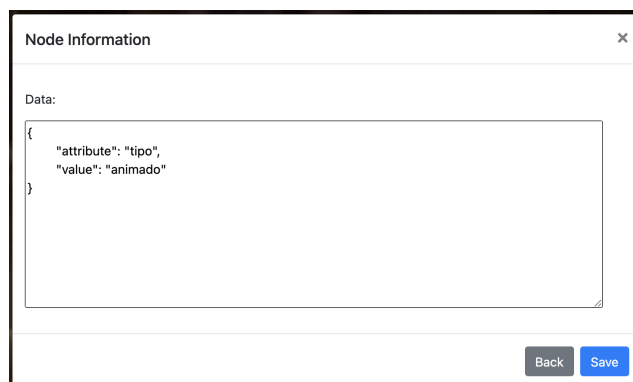
In order to show the representation of information in depth, one of the use cases of the idea presented in this paper is a tool named **Lyntax**. **Lyntax** is a compiler for a domain specific language which intends to enable the teacher to specify different kinds of sentence structures, and then, ask the student to test his own sentences against those structures. It generates *ANTLR* instructions that correspond to a specific natural language sentence based on a previously created specification. When processed, an *ANTLR* file is generated, which is then processed in order to create a visual syntax parse tree of the input (in 2D), allowing for a better understanding of the sentence. The syntax tree generated can then be post-processed and used to create an AR graph, allowing a even more visual and interactive way of envision the syntax tree.

With **Lyntax** it was possible to verify the application of the ideas presented in which they are successful in explaining the effect of an algorithm on the visualized structure. The results of the tree generated by the **Lyntax** tool using Augmented Reality can be seen in the figure 11 For more information, the tool can be access at <https://lyntax.ep1.di.uminho.pt>.



■ **Figure 11** Example of Lyntax in AR.

Each component of the sentence can have different types of attributes in order to have a more enhanced analysis. For example, Figure 12 has a sentence written in Portuguese, in which two of the components (*sujeito* – subject and *verbo* – verbo) have a common attribute, *tipo* (type). This type can be both “animated” or “inanimated”. All of the different information that guides each node within **Lyntax** is also ported for the AR graph.



■ **Figure 12** Example of Lyntax component attribute.

The prototype enhances **Lyntax**'s syntax tree, as it allows the user to view each component attributes in real-time. The user is able to analyze the sentence while visualizing the attributes that flow within the syntax tree. This takes advantage of the best practices of Augmented Reality and explores the depth of information in the tree.

5 Conclusions

The prototypes developed in the context of the project here reported, for building trees of words in Augmented Reality, made it possible to apply and evaluate the proposed visualization technique. It should be emphasized the importance of a valid strategy to show a solution to a problem in 3D. The information is presented to users in a simple and objective way, to favor the abstraction of relevant information. For many years we have demanded for appropriate educational tools and now we have the possibility of select and use visualization/animation tools that use 3D and AR to understand complex concepts.

With the referred prototypes, it was possible to corroborate the improvements resulting from the use of the third dimension (depth) to visualize hierarchical structures. Besides, the use of hyper-textual interfaces contributes significantly to the reduction of visual disorder with a large volume of information to be presented. Thus, the proposed AR system allows the user to navigate, interact and search for information within the structure, is an important alternative to traditional methods providing an effective way to comprehend the effect of the algorithms on a data structure. As suggestions for future work, we intend to discuss the adaptation of the proposed technique to the visualization of different kinds of structures, such as *ConeTree*, computer networks or networking systems, complex chemical molecules, or other type of hierarchical structure. Another idea for future work is the integration of this tool with systems that manipulate knowledge bases. With AR it is possible to create better analogies in order to visualize complex hierarchical structures, such as a list of geographic and temporal information. In addition, the efficiency of visualization of the proposed technique will be investigated, as well as the scalability of the approach. At last, it is intended to explore the Learning Resource in an educational environment to verify its impact when compared to traditional methods.

References

- 1 M.O. Almeida. Uma ferramenta para mineração visual de dados usando mapas em árvore e suas aplicações. *Salvador: Universidade Salvador*, 2003.
- 2 Ronald T Azuma. A survey of augmented reality. *Presence: Teleoperators & Virtual Environments*, 6(4):355–385, 1997.
- 3 C. Bachmaier. A radial adaptation of the sugiyama framework for visualizing hierarchical information. *IEEE transactions on visualization and computer graphics*, 13(3):583–594, 2007.
- 4 Mackinlay Card. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.
- 5 Stuart K Card, Jock D Mackinlay, and Ben Shneiderman. Using vision to think. *Readings in information visualization: using vision to think*, pages 579–581, 1999.
- 6 C. M. D. S. Freitas and al. et. Introdução à Visualização de Informação. *Revista de Informática Teórica e Aplicada*, 2:143–158, 2001.
- 7 E Bruce Goldstein and James Brockmole. *Sensation and perception*. Cengage Learning, 2016.
- 8 B. Johnson and B. Shneiderman. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the 2nd conference on Visualization'91*, pages 284–291. IEEE Computer Society Press Los Alamitos, CA, USA, 1991.

- 9 Martin Luboschik, Philip Berger, and Oliver Staadt. On spatial perception issues in augmented reality based immersive analytics. In *Proceedings of the 2016 ACM Companion on Interactive Surfaces and Spaces, ISS '16 Companion*, page 47–53. Association for Computing Machinery, New York, NY, USA, 2016. doi:10.1145/3009939.3009947.
- 10 T.W. Pratt. A hierarchical graph model of the semantics of programs. In *Proceedings of the May 14-16, 1969, spring joint computer conference*, pages 813–825. ACM, 1969.
- 11 J. Rekimoto and M. Green. The information cube: Using transparency in 3d information visualization. In *Proceedings of the Third Annual Workshop on Information Technologies & Systems (WITS'93)*, pages 125–132. Citeseer, 1993.
- 12 Tobias Sielhorst, Christoph Bichlmeier, Sandro Michael Heining, and Nassir Navab. Depth perception—a major issue in medical ar: evaluation study by twenty surgeons. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 364–372. Springer, 2006.
- 13 H. Song, Y. Qi, L. Xiao, T. Zhu, and E.P. Curran. LensTree: Browsing and Navigating Large Hierarchical Information Structures. In *Artificial Reality and Telexistence—Workshops, 2006. ICAT'06. 16th International Conference on*, pages 682–687, 2006.
- 14 Edward R. Tufte. Theory of Data Graphics. In *The Visual Display of Quantitative Information*, volume 2, pages 91–191, 2007.

Experiments on PR-Based Gamification

Alberto Simões   

2Ai, School of Technology, IPCA, Barcelos, Portugal

Ricardo Queirós   

CRACS - INESC-Porto LA, Portugal

uniMAD - ESMAD, Polytechnic of Porto, Portugal

Abstract

This article documents some experiments on teaching a class on a Master Degree Program using a different perspective on gamification. Instead of winning badges or getting achievements, students earn classification points. This allows them to work as hard as they are willing, having in mind their current classification and how far they can reach. In the specific experiment being reported, students can earn final grade points with pull requests to a shared class project. The article describes the details of the experiment, extrapolates on different ideas for implementing this in other classes, and concludes with the pros and cons of such approach for student evaluation.

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

Keywords and phrases computer education, gamification on class, GIT

Digital Object Identifier 10.4230/OASICS.ICPEC.2021.16

Category Short Paper

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).

1 Introduction

Computer Science teaching is a complex task, not just on the first years, but also in advanced courses, like Master Degrees. Usually, teachers tend to use hybrid evaluation methodologies, comprised of theoretical and practical parts. Both types require different strategies for engaging students. In this contribution we are not dealing with theoretical evaluation, but only on how to engage students on practicing writing code.

A common approach for evaluating the student competences on writing code is to ask them to develop some kind of solution. Sadly, for some courses, that is not easy. For instance, when the solution that makes more sense to request to students is a large application, and just a part is related to the goals of the curricular unity. To remedy this problem, teachers require students to work in groups, dividing the work load. This leads to other kind of issues, like non balanced competences among the group elements, making it hard to properly evaluate each one independently.

This document described an experiment in a Master Degree on Digital Games Development Engineering, at the School of Technology of the Polytechnic Institute of Cávado and Ave, in Barcelos, Portugal, especially in the course of “Game Engines,” where the students learn how to structure and build from scratch the basic functionalities for a game engine. This is a 45 hours course, of the first year of this Master's degree.

This document is structured as follows: Section 2 discusses on different gamification approaches already discussed by other authors; Section 3 explains the methodology used in this experiment; Section 4 does an analysis of the experiment, pointing out for different areas where this kind of gamification can lead; finally Section 5 draws some conclusions.



© Alberto Simões and Ricardo Queirós;
licensed under Creative Commons License CC-BY 4.0

Second International Computer Programming Education Conference (ICPEC 2021).

Editors: Pedro Rangel Henriques, Filipe Portela, Ricardo Queirós, and Alberto Simões; Article No. 16;
pp. 16:1–16:10



OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2 Related Work

Gamification can be defined as the use of game elements in non-game contexts [5]. In practice, game elements can be considered as a conceptual toolkit which can be applied in a system in order to motivate certain behaviors. Among all relevant game elements, the classic PBL triad (Points, Badges, and Leader-boards) is often adopted in most of the gamification projects to reward/notify individuals upon the accomplishment of specific tasks. The primary reason for their use is its simplicity (common sense in understanding their rules) and its effectiveness to achieve early motivation.

2.1 Motivation

Gamification is no longer a buzzword. Nowadays, gamification plays a central role and is being considered as the solution for many issues related with lack of retention and demotivation in education and business domains. In order to successfully apply gamification it is crucial to understand the level and type of motivation that we want individuals to achieve using our system.

Motivation can be defined as the core element which drives individuals to accomplish a task. Beyond the amount of motivation that is necessary to achieve in order to be able to complete a specific task, it is important to distinguish and manipulate their two major flavors: *extrinsic* and *intrinsic motivation* [22].

Extrinsic motivation is gained by pursuing tangible rewards acquired after successfully doing an activity (e.g., rankings, levels, points, badges, awards, financial incentives) [19]. Intrinsic motivation is acquired by doing an activity for the own purpose of the individuals inherent satisfaction and, most of the time, as an opportunity to learn or recycle their potentials. In this context, it is important to foster cooperation, competition, self-esteem, ego, sense of belonging and love [19].

Most of the time, intrinsic motivation offers long-term and high-quality behavioral engagement, whereas extrinsic motivation is relatively less effective. Nevertheless, extrinsic motivation is still necessary in scenarios where the activity is itself neither engaging nor rewarding to the individual. In this case, external rewards are delivered to foster participation. Yet it should not be overused, since excessive external rewards might impair individuals spontaneous interest in the activity [4].

Based on these facts, the balance between both types of motivation should be encouraged and applied in the gamification construction process. Although advised, this balance is very difficult to achieve. The majority of the gamified systems can involve users in the first weeks, but then users lose motivation either due to the game mechanics' routine, or sometimes due to lack of transparency in the application's point system. This routine or lack of confidence leads the user to feel discredited and to abandon the application. Thus, the perfect symbiosis of these two types of motivation perpetuates the users' confidence, leading them to be able to complete the tasks in order to be rewarded and, at a later stage, to achieve an inner satisfaction that goes beyond any reward and is more related to self-improvement and the desire to be better [5].

2.2 Gamification in the Computer Programming domain

Gamification is being widely used in education and business contexts. In the education realm, gamification is often used to facilitate the learning process of tedious and/or complex subjects. One of these subjects is computer programming.

It is a fact that learning how to code is a complex process [11, 21]. Early systems started by trying to mimic traditional teaching methodologies to the virtual world with few success. In fact, those systems were characterized by the excessive focus on the language syntax and the availability of programming exercises without full coverage of the course curricula, unbalanced in terms of complexity and not suitable for heterogeneous classes with students with different profiles. In the beginning of the century, most systems included automatic evaluation systems to foster practice. However, we continue to assist to students' lack of motivation and the consequent poor grades and dropouts. In the last decade, gamification began to be used to circumvent this scenario and several programming learning platforms appeared.

These platforms can be grouped in three major categories:

- Massive Open Online Courses (MOOC),
- Competition Systems, and
- Serious Games.

The next subsections describes the most notable examples of the three categories. Other categories were left out such as code playgrounds¹ and sandboxes. The main reason for discarding those categories is the fact that their main focus is not learning and for their lack of game elements.

2.2.1 Massive Open Online Courses

Currently, there is a large number of open online programming courses available on the web such as Coursera², Udacity³, edX⁴, Codecademy⁵, Khan Academy⁶, and Free Code Camp⁷. Most of these platforms offer a wide variety of learning material from top universities' courses, with the possibility, at the end of the course, of getting paid certificates.

These platforms typically use gamification to attract learners, adopting elements which go beyond the PBL triad such as different levels, progress indicators and experience points. For example, Khan Academy uses badges and progress tracking to engage students to enlist and complete courses. Codecademy uses levels to organizes lessons and progress indicators to notify learns of their current learning status.

In the context of educational institutions, the most widely used approach to create gamified open online courses is to rely on an Learning Management System (LMS) with game mechanics. LMSs were created to deliver course contents, and collect assignments of the students. However, many of them evolved to provide more engaging environments resorting to gamification. Some of the most notable examples are Academy LMS⁸, Moodle⁹, and Matrix¹⁰ which include badges, achievements, points, and leader-boards. Moodle also has several plugins which offer a variety of other gamification elements [17].

¹ CodePen (<https://codepen.io>), CodeSandbox (<https://codesandbox.io/>), PlayCode (<https://playcode.io/>), JSFiddle (<https://jsfiddle.net/>), SoloLearn (<https://www.sololearn.com/>).

² Available at <https://www.coursera.org/>.

³ Available at <https://www.udacity.com/>.

⁴ Available at <https://www.edx.org/>.

⁵ Available at <https://www.codecademy.com/>.

⁶ Available at <https://www.khanacademy.org/>.

⁷ Available at <https://www.freecodecamp.org/>.

⁸ Available at <https://academy-lms.com/>.

⁹ Available at <https://moodle.org/>.

¹⁰ Available at <https://www.matrixlms.com/>.

One of the most relevant examples is Enki [17] which is a tool that blends learning with assessment and gamification,. In Enki, the content is presented in several forms as well as delivering programming assignments with automatic feedback, while allowing anyone to create courses freely.

2.2.2 Competition Systems

It is part of the human condition to be competitive. Despite some dangers associated with learning based competition systems [12], it has proven to be an effective technique to stimulate students to exceed their capabilities [14, 3]. In this realm, programming contests are becoming more popular among learners [25]. This section presents some platforms and tools that use competition as a way to engage students in learning programming.

The most notable example of competition systems are the programming contests which can be defined as environments with a set of competitive activities where individuals or teams strive to find solutions for a specific set of problems. In these contests, the evaluation of submissions are handled by special components (often called automatic judges) allowing participants to make as many submissions as they want during competition, and learn from their mistakes. In this way, “losers” actually win knowledge which may compensate the negative effects typically associated with competitive learning activities [20].

The most popular automatic judges are DOMJudge [8], PC² [1], PKU JudgeOnline [18], and Mooshak [13]. Despite being used primarily to support programming contests, currently the majority of them are capable of providing rich and immediate feedback to students, allowing the motivation by using timed challenges and leader-boards [9].

There are several platforms that can be used to compete or train as part of the recruitment process for top technology companies. Some notable examples are HackerRank¹¹, CodeSignal¹², Codility¹³, HackerEarth¹⁴ Assessments, TestDome¹⁵, HireVue¹⁶, DevSkiller¹⁷, iMocha¹⁸ and CodinGame For Work¹⁹. Usually so-called as Tech Recruiting Platform or Remote Online Code Testing, these platforms offer PBL facilities and include contests which can assume different time frames (opened indefinitely, in which challenges are proposed every week/day/month, or run for a limited amount of time – 48 hours). These contests enable users to increase/decrease their rating, win medals (which are given to top solutions), and cash prizes or, even, jobs at top technology companies.

2.2.3 Serious games

A serious game is a game designed for a primary purpose other than pure entertainment [6]. Currently, serious games are applied in several domains including healthcare, well-being, advertisement, and education. In the later, serious games are used to mitigate the difficulties found when learning a specific subject, either because it is complex or boring. In this scope, serious games are being applied in computer programming learning in order to explain

¹¹ Available at <https://www.hackerrank.com/>.

¹² Available at <https://codesignal.com/>.

¹³ Available at <https://www.codility.com/>.

¹⁴ Available at <https://www.hackerearth.com/>.

¹⁵ Available at <https://www.testdome.com/>.

¹⁶ Available at <https://www.hirevue.com/>.

¹⁷ Available at <https://devskiller.com/>.

¹⁸ Available at <https://www.imocha.com.my/>.

¹⁹ Available at <https://www.codingame.com/>.

programming concepts or to foster the practice of skills that, for some reason, are harder to assimilate.

Many studies were made aiming to identify requirements for an educational game to promote the learning of problem-solving techniques in introductory programming [26].

A survey [16] including 49 serious games analysed the most common features, and those which may increase the game's adoption. In this scope, the importance of fostering collaboration and competition within the game and the game's coverage based on the ACM reference curriculum for Fundamental Concepts in Programming were the most cited missing features. Other conclusion of this study was the weak attention to the adoption of the best accessibility and inclusion practices.

A recent survey [23] analysed 41 games and founded that most of them emphasize mechanics and dynamics rather than aesthetics, making in general, the games tiring and boring. Other weak point was that the majority of the games analysed are not directed linked to the undergraduate level and the poor coverage of the items in the ACM curriculum. As a main conclusion of this study, authors recommended considering the student's previous skills and knowledge and providing automatic and rich feedback when specific error events occurs.

As some notable examples, selected without any criteria, are NoBug's SnackBar, Wu's Castle and Robocode:

- NoBug's SnackBar [26] is a serious game to support the learning of basic computer programming. It is a blocks-based environment including also resources that allow the teacher to follow the student's progress and customize in-game tasks.
- Wu's Castle [7] aims to teach basic programming concepts such as loops and arrays by presenting the learner with multiple-choice questions or fill-in blanks. The feedback consists of displaying an avatar executing the code.
- Robocode [2, 10] challenges the student to develop a complete software agent capable of defeating all the opponents in the arena in order to practice object-oriented programming and structured programming.

3 Gamification Methodology

As described previously, the experiment was performed on a Master Degree course, with 15 students. The topic of the course is game engine development, and the main goals are to understand the different components of a game engine, including but not limited to managing the input from the player, perform the rendering tasks (sprites, cameras, models), simulate physics (namely collision) and play sound.

In order to allow students to understand these components and how they interact together, one of the objectives is to have students developing modules, that can be put together, and implement a simple game. Given the students background, the used technologies were the C# programming language, and the MonoGame²⁰ framework.

As to students motivation, badges or achievements, by themselves, is not enough for engagement. While most studies defend that these kind of rewards motivate students, there is the lack of analysis of long-term effect of gamification. Accordingly with some authors [24, 15], studies are performed during short periods of time, and there is no real analysis of the benefit of virtual rewards on situations when "players" have little time and no direct real reward for they effort.

²⁰<https://www.monogame.net/>

16:6 Experiments on PR-Based Gamification

Most of the students are employed, as the master degree runs in an after-work basis. Their time for studies is quite limited, and therefore, their main goal is to complete the degree, and if possible, with an interesting grade. With this in mind, the defined methodology awards grade points to the students, in a way students can accumulate them. At the end, the amount of points the students are able to gather is their final grade²¹ (of course, with a possible truncation on the maximum grade). This scenario was described to the students in the first class, allowing them to understand the mechanics.

This gamification approach has a direct impact on their grade, as not just the collection of badges or achievements. Of course we can award a grade for each obtained badge or achievement, but that would be a similar experience to the one being described.

The methodology to assign the grade points to the students was aligned to the completion of tasks, hidden as the creation of pull requests. Thus, during classes, specific tasks were open in an issue tracker, and classified with the grade that would be awarded to the student resolving the issue. In this specific situation, a private repository from GitLab²² was used.

During the classes concepts were discussed and a basic structure of the code needed to handle the discussed functionalities were added to the repository. At the end (or after the class), the teacher opens a set of issues in the GitLab interface. Each issue is a functionality needed for the common class project, and an amount of grade points is defined as the reward for that task.

Different tasks might have different reward amounts. Simpler tasks were ranging from 1 to 3 points, while some more complex tasks could award up to 6 points. To guarantee students do not choose all the simple tasks for themselves, after a task being assigned the student is unable to get another task until his current assignment is complete.

To complete an assignment the student needs to develop the code and prepare a pull request (named *merge request* in GitLab). The teacher validates the code, suggesting corrections or just accepting the solution. Every time a pull request is merged, the amount of points defined in the task are credited to the student. A possibility would be awarding only partially, if the solution is not the best. Nevertheless, during the experiment we are describing, that was not the case.

After one task is done, the teacher can create new issues to complete the code from a previous student with new features, or merging two different modules developed previously into a single functionality. This means that, as the classes evolve, the complexity of the implementation increases.

4 Discussion and Evaluation

While in the previous section the gamification methodology was presented, in this section we will discuss the pros and cons found during the experiment, following a SWOT methodology.

4.1 Weaknesses

Unfortunately there are some drawbacks on adopting this methodology. We present some of them now:

- As expected, this approach requires more work from the teacher. While teachers already need to tutor their students when performing any other project, this kind of task requires

²¹ In Portugal, student grades are in the interval 0–20.

²² <https://gitlab.com>

more effort, not just because that each student will be working on a different problem, but also because from time to time a new set of issues needs to be published, allowing students to continue their work.

At the same time, if the class is too big, it might be not feasible to have some many parallel tasks at once, forcing some students to be idle, waiting for others to submit their pull requests, in order to be able to get their own assignments.

- The way students solve a specific task might not be the best one for other features that will be coded on top of that. While the teacher might do code validation, and request some changes, to force a specific approach for solving a problem might not be desired – it is useful to let the students understand what path to go, and deal with the problems that might arise from there. Thus, sometimes, creating new issues that require to deal with code developed by a previous student might require some extra work. In this situations, our approach was to create intermediate tasks to transform the code in a way it gets suitable for the next task.

4.2 Strengths

During the experiment we noticed some advantages on this methodology:

- Students are able to work as much as they like, and at each step, they know exactly their current grade. While some students might be comfortable with a low but positive grade, other students will keep fighting to get a higher grade. As the effort to raise their score is manageable, and known *a priori*, they can decide which tasks they want to perform, and choose their own path.
- Students with low coding skill usually struggle to develop a complex project. If the teacher is able to propose issues of different complexity levels, with a low point reward, good students will not opt for those tasks, as they would need to do many more simple tasks to get to a high final grade. But, for students with low coding skills, these are an opportunity to gather some points. They might need to solve a larger number of issues, but their hard work will be rewarded with a positive grade. This is also an approach to make these students work more time, and therefore, learning better coding skills.
- Making students to develop tasks on already existing code forces them to read other people code. This is an important part of the learning process of coding. If the first tasks are easier to implement, as they are to be developed on top of the code written during a class, and therefore, explained, as new tasks are proposed students are required to look to code from their colleagues.
- While the first detected weakness is regrading bad code practice or not so versatile implementations, when a new issue for code refactoring is opened, most of the times the students that performed the original code are willing to take it, and refactor their own code, as they see that as an interesting challenge.

4.3 Opportunities

There are different opportunities for this approach, namely:

- The possibility to develop a system to track automatically the pull request scores, allowing the students to follow their grade evolution.
- This kind of approach, with some adaptations, might be used in other situations. While in this experiment all students were working on the same code-base, the use of generic

open-source projects for which students can contribute would also work. For instance, a class clone of the Hacktoberfest²³ could be very interesting.

4.4 Threats

We foresee some problems of this approach:

- Students can work for a minimum grade. This is not an interesting situation for the teacher, as students might just drop their attention for the remaining classes if they feel they goal was already achieved.
- Not all type of classes, even for computer science courses, can benefit of such approach, as creating different tasks for every student is not easy.
- Related to the previous threat, this methodology would not work for a high number of students, as it will not be possible, for the teacher, to create as much tasks as required to allow every student to have their opportunity.

5 Conclusions

When this experiment was conducted, the main idea was not to report about it, but to give a boost on the teaching methodology for this specific course. Therefore, unfortunately there was no evaluation on the approach by the students, and given we could not reproduce this methodology last year, given the global pandemic, currently we do not have a proper evaluation by the students.

Nevertheless, during the experiment there was two main types of students: some of the students struggled to get a positive evaluation, and stopped working; but some other found the approach interesting and got interesting grades. From this last group, not all students were experienced programmers. Some of them achieved a good grade very quickly, performing more complex tasks. Some others, with more difficulties, were able to achieve their grades by preparing simpler pull requests, but with a higher number of solved issues.

One of the drawbacks of this approach was the lack of a system to keep track of the students score. At that time, the class used a plain public GitLab server (although with a private repository). Nevertheless, as GitLab is freely available, one could install it locally, and it would be possible to develop a plugin to allow this gamification to be performed automatically, without the need for the teacher to keep track of which pull requests were solved, and their rewards.

References

- 1 Samir E. Ashoo, Troy Boudreau, and Douglas A. Lane. CSUS Programming Contest Control System (PC2), 2018. URL: <http://pc2.ecs.csus.edu/> [cited September 2020].
- 2 Esmail Bonakdarian and Laurie White. Robocode throughout the curriculum. *J. Comput. Sci. Coll.*, 19(3):311–313, January 2004.
- 3 Juan C. Burguillo. Using game theory and competition-based learning to stimulate student motivation and performance. *Computers & Education*, 55(2):566–575, 2010. doi:10.1016/j.compedu.2010.02.018.
- 4 D. Coon and J.O. Mitterer. *Introduction to Psychology: Gateways to Mind and Behavior with Concept Maps and Reviews*. MindTap Course List Series. Cengage Learning, 2012. URL: <https://books.google.sm/books?id=EYwjCQAAQBAJ>.

²³See <https://hacktoberfest.digitalocean.com/>.

- 5 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*, MindTrek '11, page 9–15, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/2181037.2181040.
- 6 Damien Djaouti, Julian Alvarez, and Jean-Pierre Jessel. Classifying serious games: the g/p/s model. *Handbook of Research on Improving Learning and Motivation through Educational Games: Multidisciplinary Approaches*, January 2011. doi:10.4018/978-1-60960-495-0.ch006.
- 7 Michael Eagle and Tiffany Barnes. Wu's castle: teaching arrays and loops in a game. In *Proceedings of the 13th annual conference on Innovation and technology in computer science education*, ITiCSE '08, pages 245–249. ACM, 2008. doi:10.1145/1384271.1384337.
- 8 Jaap Eldering, Thijs Kinkhorst, and Peter van de Warken. DOMjudge-programming contest jury system, 2011. URL: <https://www.domjudge.org/> [cited September 2020].
- 9 Pedro Guerreiro and Katerina Georgouli. Enhancing elementary programming courses using e-learning with a competitive attitude. *International Journal of Internet Education*, 10, January 2008.
- 10 Ken Hartness. Robocode: Using games to teach artificial intelligence. *J. Comput. Sci. Coll.*, 19(4):287–291, 2004.
- 11 Tony Jenkins. On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, volume 4, pages 53–58, 2002.
- 12 Alfie Kohn. *No contest: The case against competition*. Houghton Mifflin Harcourt, 1992.
- 13 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.
- 14 José Paulo Leal and Fernando Silva. Using Mooshak as a Competitive Learning Tool. In *The 2008 Competitive Learning Symposium*, 2008.
- 15 Elisa D. Mekler, Florian Brühlmann, Klaus Opwis, and Alexandre N. Tuch. Do points, levels and leaderboards harm intrinsic motivation? an empirical analysis of common gamification elements. In *Proceedings of the First International Conference on Gameful Design, Research, and Applications*, Gamification '13, page 66–73, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2583008.2583017.
- 16 Michael A. Miljanovic and Jeremy S. Bradbury. A review of serious games for programming. In Stefan Göbel, Augusto Garcia-Agundez, Thomas Tregel, Minhua Ma, Jannicke Baalsrud Hauge, Manuel Oliveira, Tim Marsh, and Polona Caserman, editors, *Serious Games*, pages 204–216, Cham, 2018. Springer International Publishing.
- 17 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.
- 18 Xu Pengcheng, Ying Fuchen, and Xie Di. PKU JudgeOnline, 2013. URL: <http://poj.org/> [cited September 2020].
- 19 Kalliopi Rapti. Increasing motivation through gamification in e-learning. *The Journal for Open and Distance Education and Educational Technology*, 7, June 2016. doi:10.12681/icodl.640.
- 20 Miguel A. Revilla, Shahriar Manzoor, and Rujia Liu. Competitive learning in informatics: The UVa online judge experience. *Olympiads in Informatics*, 2:131–148, 2008.
- 21 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.
- 22 Richard M. Ryan and Edward L. Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary Educational Psychology*, 25(1):54–67, 2000. doi:10.1006/ceps.1999.1020.

16:10 Experiments on PR-Based Gamification

- 23 M. Shahid, A. Wajid, K. U. Haq, I. Saleem, and A. H. Shujja. A review of gamification for learning programming fundamental. In *2019 International Conference on Innovative Computing (ICIC)*, pages 1–8, 2019. doi:10.1109/ICIC48496.2019.8966685.
- 24 Stefan Stepanovic and Tobias Mettler. Gamification applied for health promotion: does it really foster long-term engagement? A scoping review. In Peter M. Bednar, Ulrich Frank, and Karlheinz Kautz, editors, *26th European Conference on Information Systems: Beyond Digitization - Facets of Socio-Technical Change, ECIS 2018, Portsmouth, UK, June 23-28, 2018*, page 50, 2018. URL: https://aisel.aisnet.org/ecis2018_rp/50.
- 25 Andrew Trotman and Chris Handley. Programming contest strategy. *Computers & Education*, 50(3):821–837, 2008. doi:10.1016/j.compedu.2006.08.008.
- 26 Adilson Vahldick, Paulo Roberto Farah, Maria José Marcelino, and António José Mendes. A blocks-based serious game to support introductory computer programming in undergraduate education. *Computers in Human Behavior Reports*, 2:100037, 2020. doi:10.1016/j.chbr.2020.100037.

User Experience Evaluation in a Code Playground

Ricardo Queirós   

CRACS - INESC-Porto LA, Portugal

uniMAD - ESMAD, Polytechnic of Porto, Portugal

Mário Pinto  

uniMAD - ESMAD, Polytechnic of Porto, Portugal

Teresa Terroso  

uniMAD - ESMAD, Polytechnic of Porto, Portugal

Abstract

Learning computer programming is a complex activity and requires a lot of practice. The viral pandemic that we are facing has intensified these difficulties. In this context, programming learning platforms play a crucial role. Most of them are characterized by providing a wide range of exercises with progressive complexity, multi-language support, sophisticated interfaces and automatic evaluation and gamification services. Nevertheless, despite the various features provided, others features, which influence user experience, are not emphasized, such as performance and usability. This article presents an user experience evaluation of the LearnJS playground, a JavaScript learning platform which aims to foster the practice of coding. The evaluation highlights two facets of the code playground: performance and a usability. In the former, lab and field data were collected based on Google Lighthouse and PageSpeed Insights reports. In the later, an inquiry was distributed among students from a Web Technologies course with a set of questions based on flexibility, usability and consistency heuristics. Both evaluation studies have a twofold goal: to improve the learning environment in order to be officially used in the next school year and to foster the awareness of user experience in all phases of the software development life-cycle as a key facet in Web applications engagement and loyalty.

2012 ACM Subject Classification Applied computing → Computer-managed instruction; Applied computing → Interactive learning environments; Applied computing → E-learning

Keywords and phrases programming learning, code playground, programming exercises, user experience

Digital Object Identifier 10.4230/OASICS.ICPEEC.2021.17

Category Short Paper

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).

1 Introduction

The JavaScript language is no more confined to the browser. Nowadays the language gravitates on almost all platforms, assuming a crucial role in distributed and embedded systems. This success came to underpin its use not only by IT companies but also in the first years in computer science related courses.

In this context, LearnJS emerged as a playground for practicing JavaScript coding. There are several playgrounds worldwide. Many of them adopting the paradigm one-size-fits-all paradigm with sophisticated interfaces and offering a large number of exercises with progressive complexity and covering all the typical topics within the curricula of a computer programming course with multi-language support. In order to engage users and accelerate



© Ricardo Queirós, Mário Pinto, and Teresa Terroso;
licensed under Creative Commons License CC-BY 4.0

Second International Computer Programming Education Conference (ICPEEC 2021).

Editors: Pedro Rangel Henriques, Filipe Portela, Ricardo Queirós, and Alberto Simões; Article No. 17; pp. 17:1–17:9
OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

17:2 User Experience Evaluation in a Code Playground

their extrinsic motivation, some of these platforms include evaluation services with automatic feedback and gamification elements such as leaderboards, achievements, badges and points.

Despite all these features, there are others that aren't enhanced or, sometimes, even neglected such as those that influence more directly the user experience. Therefore, the main requirement in the design of LearnJS was to foster the user experience (UX) in the perspective that having happy users benefit their involvement and, consequently, their loyalty. In order to achieve this requirement, at the platform design phase, two facets of the user experience were addressed: performance and usability. The former was enhanced with the creation of the playground as a Single Page Application with a small number of round trips to the server. The later, was considered by adopting a UI (User Interface) framework in order to build rich and engaging user experiences composed by crafted components from the Material Design specification.

Despite all these precautions, it is necessary to assess whether they have had a positive effect on the user experience. This article presents a performance and an usability study in the LearnJS playground. The performance study was based on lab and field data obtained from Google Lighthouse and PageSpeed Insights tools. The second is based the perception that users had on the use of the playground. In order to collect these perceptions a usability inquiry was delivered to first- and second-years students of the Degree in Technologies and Information Systems at ESMAD (Superior School of Media Arts and Design) from the Polytechnic Institute of Porto (P.PORTO). The inquiry's questions were based on Nielsen's heuristics such as flexibility, usability and consistency.

These studies aim to achieve two objectives: to improve the user experience on LearnJS, and to raise the awareness of all the workflow agents in the software life-cycle development about the importance of UX in the development of enjoyable virtual learning platforms.

The rest of the article is structured as follows: in section 2 a brief state of the art on code playgrounds and heuristics evaluation models are presented. Section 3 presents LearnJS, more precisely, its architecture, main components, and the playground GUI. In section 4, two types of evaluation in the LearnJS playground were made: a performance evaluation based on Google performance tools reports and an usability evaluation based on the analysis of a survey filled by students of a Web Technology course. Finally, the main contributions of this work are presented and the next steps regarding the evolution of the playground are shared.

2 Code Playgrounds

A variety of code playgrounds have appeared over the years. These type of tools help students to practice a programming language without the need to install any specialized editor or IDE in their computers. The next subsections present a brief survey on code playgrounds and highlight the heuristic evaluation as the main method for evaluating the usability of interactive system such as these types of playgrounds.

2.1 State of the Art

Nowadays, there are several Web applications aiming to foster coding skills. These applications are often coined with several names such as interactive online courses, massive open online course, coding bootcamp tools, code playgrounds, code interview prep platforms, and many others.

Regardless of the names, all of these platforms have something in common – they all encourage the practice of coding. Since solving exercises and receiving feedback on their resolution is considered to be one of the most efficient way to learn programming, these

types of tools are important strategies for students to autonomously consolidate all the programming topics in an informal, enjoyable and interactive way.

Within this scope there are several tools that stand out such as freeCodeCamp, Coursera, codecademy, edX, Udemy, Udacity, W3schools, SoloLearn, Hackr.io, coderbyte, codewars, hackerrank, codingame, khan academy, qvault, treehouse, lynda, futurelearn, codeasy, zenna academy, and others.

There are several comparative studies [4, 5, 6] analyzing the differences between these tools using several predefined criteria. One of such studies [6], authors compare these code playgrounds based on set of predefined criteria such as edition, evaluation, gamification and interoperability.

In addition to the common objective, many of these platforms share a similar graphic structure that can be grouped into the following components:

- collection of exercises - list of exercises organized by modules and with increasing complexity. In some environments the availability of modules and/or exercises depends on the success in solving the previous exercises;
- statement - description of the challenge in several formats and languages. Beyond the challenge, the text often includes a set of example input/output tests that the student can use for self-evaluation purposes;
- editor - specialize editor embedded in the Web page and properly configured for the supported languages. The main features are syntax highlighting, auto-completion, snippets injection and presentation of syntactical errors;
- console - enumeration of errors, warnings, hints in order to notify students on their actions;
- tests - static and dynamic tests. The former is the base for analyzing the code without running it and detecting many issues from keyword detection to applied code style. The later is based on input/output tests where the student's code is executed with the given input and the output generated upon execution is, then, compared with the given output;
- related resources - dependence resources that are suggested to the students to consolidate the current topic (e.g. similarly exercises, video tutorials);
- social - this component gathers several features that allow the interaction with the community such as forums, code sharing facilities, pair programming, and many others;
- gamification - game elements aiming to motivate students on their progress in the course. They can be materialized as several widgets such as a ranking, a set of badges, or even experience points earned by the students that can unblock levels, permissions or be used later for internal trading.

2.2 Heuristic Evaluation

Heuristic Evaluation (HE) is one of the most widely used methods for evaluating the usability of an interactive system.

Since the time that Schneiderman [8] presented his well-known “Eight Golden Rules of Interface Design”, and passing for the popular Nielsen’s Ten general principles for interaction design [3] or Tognazzini’s First Principles of Interaction Design [9], there were several studies presenting new sets to help UI designers, developers and other experts in their goal of enhancing usability. Despite the number, most of these studies boil down to modifying Nielsen’s list and/or adding new principles to evaluate specific aspects not covered. A complete review of several sets of usability heuristics created for specific domains (not only programming) by different authors can be found in [7].

17:4 User Experience Evaluation in a Code Playground

Nowadays, almost everybody refers to evaluate usability or UX based on the Nielsen's heuristics list. Recent studies use Nielsen's or Tognazzini's lists as the most accurate models [1]. Other studies present hybrid solutions joining both models with the premise that Nielsen's list needed something more specific to be useful [1].

In our case, we decided to take Nielsen's list to do this present work. According to Nielsen the practical acceptability of a system includes factors such as usefulness, cost, reliability and interoperability with existing systems. The usefulness factor relates the utility and usability offered by the system. Utility is the capacity of the system to achieve a desired goal. As the system perform more tasks, more utility he has. Usability is defined by Nielsen as a qualitative attribute that estimates how easy is to use an UI. He mentions five characteristics involved in the concept of usability:

- ease of learning - the system should be easy to learn so that the user can start doing some work with the system;
- efficiency - the system should be efficient to use, so after the user learns the system, a high level of productivity is possible;
- memorability - the system should be easy to remember so that the casual user is able to return to the system after a period without using it, without requiring to learn it all over again;
- errors - the system should prevent the user from committing errors as should deal with them gracefully and minimizing the chance of occurring catastrophic errors;
- satisfaction - the system should be pleasant to use so that users become subjectively satisfied when using.

3 LearnJS

LearnJS is a Web playground which enables anyone to practice the JavaScript language [6]. The LearnJS Playground is a Web-based component which will be used by learners to browse learning activities and interact with the compound resources. Here students can see videos or PDF's of specific topics and solve exercises related with those topics with automatic feedback on their resolutions.

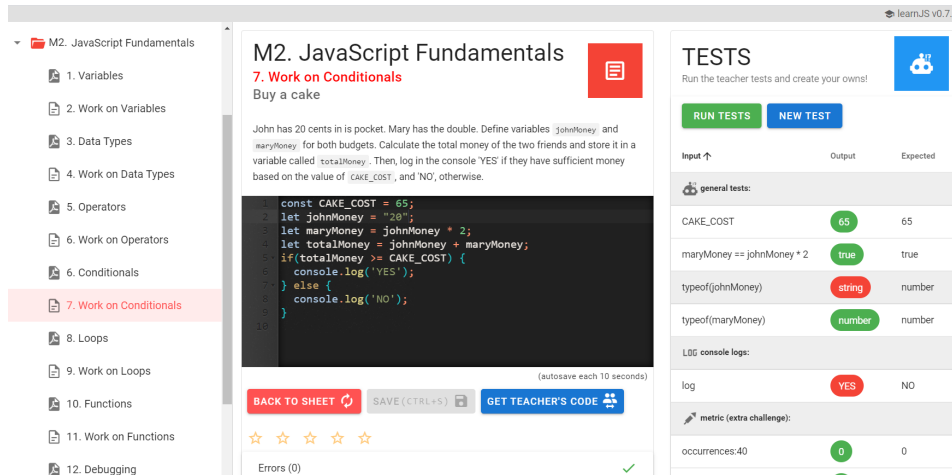
Currently, the playground has two main components:

1. Editor: allows students to code their solutions in an interactive environment;
2. Evaluator: assess the student's code based on static and dynamic analyzers.

For the Editor component, the playground uses Ace (maintained as the primary editor for Cloud9 IDE) which can be easily embedded in any web page and JavaScript application. The editor is properly configured for the JavaScript language and supports the Emmet toolkit for the inclusion of dynamic JavaScript snippets. Ace editor can display errors on the editor itself but does not handle language dependencies. A parser needs to be used to detect errors and determine their positions on the source file. There are several tools that can improve code quality. One of such cases is code linters. Linters (e.g JSLint, JSHint) can detect potential bugs, as well as code that is difficult to maintain. These static code analysis tools come into play and help developers spot several issues such as a syntax error, an unused variable, a bug due to an implicit type conversion, or even (if properly configured) coding style issues. LearnJS uses JSHint to accomplish this behavior. While static code analysis tools can spot many different kinds of mistakes, they can not detect if your program is correct, fast or has memory leaks. For that particularly reason, LearnJS combines JSHint with functional tests (based on test cases). For this kind of tests, and since the code is written in JS and the context is the browser, we use a simple approach by iterating all the case tests and

applying the eval function for tests injection. Both analyzers (linter and Test Case runner) are subcomponents of the LearnJS evaluator component that runs exclusively on the client side. This approach avoids successive round-trips to the server which affects negatively the user experience.

At this moment, a simple running prototype (version 0.7.7) is available at <https://rqueiros.github.io/learnjs/>. Figure 1 shows the front-end GUI of the playground.



■ **Figure 1** LearnJS playground GUI.

4 User Experience Evaluation

This section presents the two types of evaluation that were performed in the LearnJS playground: performance and usability.

4.1 Performance Evaluation

Nowadays, Web performance is not just a technical concern: it affects everything from accessibility to usability. Even if you look to the software development workflow, design decisions should be aware of their performance implications. There are a lot of examples that a single second spared in the page load of a company site can increase search engine traffic, sign-ups and even the average annual revenue. Thus, it is unanimous that performance is an important factor to bear in mind and it should be measured, monitored and refined continually.

However, the growing complexity of the web poses new challenges that make performance relative. For instance, a site might be fast for one user (on a fast network with a powerful device) but slow for another user (on a slow network with a low-end device) or even a site might appear to load quickly but respond slowly to user interaction.

So when talking about performance, it's important to be precise and to refer to performance in terms of objective criteria that can be quantitatively measured. These criteria are known as metrics. There are a lot of metrics from the first byte received by the browser to the time when the page is ready to be interactive. Also there are a lot of tools to measure those metrics. In order to overcome this dispersion, Google launched the Web Vitals initiative to simplify the tools and metrics landscape, and help developers focus on the metrics that

17:6 User Experience Evaluation in a Code Playground

matter most. In this realm, tools that assess Web Vitals compliance should consider a page passing if it meets the targets at the 75th percentile for all of the following three metrics:

- Largest Contentful Paint (LCP): measures the time from when the page starts loading to when the largest element is rendered on screen;
- First Input Delay (FID): measures the time from when a user first interacts with your site (e.g. when they click a link, tap a button) to the time when the browser is actually able to respond to that interaction;
- Cumulative Layout Shift (CLS): measures the cumulative score of all unexpected layout shifts that occur between when the page starts loading and when its life-cycle state changes to hidden.

These metrics are generally measured in one of two ways: in the lab using tools to simulate a page load in a controlled environment; in the field on real users actually loading and interacting with the page.

In order to evaluate the LearnJS playground performance, a field analysis was performed using the Google PageSpeed Insights (PSI). Table 1 presents the performance results for six metrics (including the three Web Vitals metrics) organized in the two well known target devices: mobile and desktop. Although the results are presented for both platforms, for the LearnJS scope, desktops are the most important as they will be those used by students in solving programming exercises.

■ **Table 1** LearnJS performance report.

	FCP ¹	SI ²	LCP	TTI ³	TBT	CLS
Mobile	7.3s	7.3s	12.3s	9.1s	450ms	0.011
Desktop	1.5s	1.5s	2.1s	1.7s	60ms	0.001

¹FCP = First Contentful Paint

²SI = Speed Index

³TTI = Time To Interactive

PSI incorporates data from the Chrome User Experience Report (CrUX) to display data on the actual performance of a page. This data represents an aggregate view of all page loads over the previous 28-day collection period. However, since LearnJS was put online very recently, CrUX doesn't have enough real speed data for this source. Therefore, data comes from the Google LightHouse that gives performance insights from a controlled environment using simulated throttling.

PSI classifies field metric values as good/needs improvement/poor by applying specific thresholds. The overall score for LearnJS was 81 points (0 to 100). It falls in the "needs improvement" category. Looking now at the mobile metric values and stressing the Web Vitals metrics, one can conclude the following:

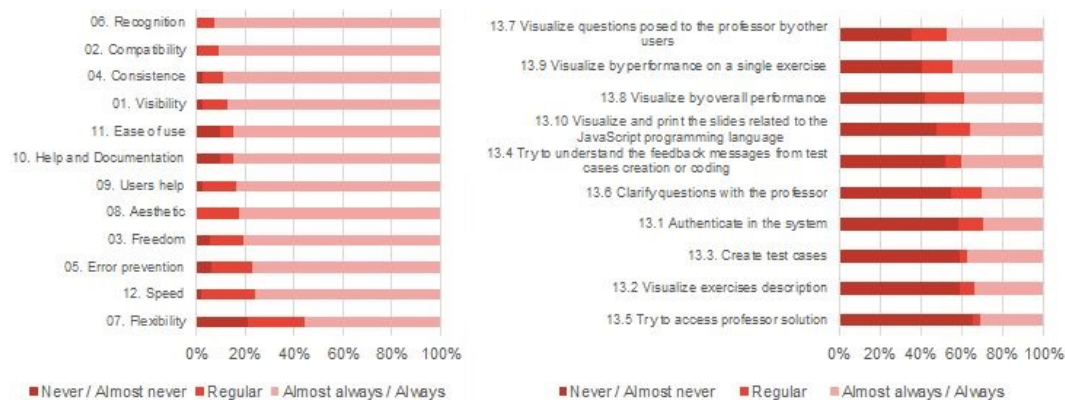
- LCP (2.1s) - this value was expected since in the SPA paradigm most of the resources are load in the first request. In order to mitigate this score, we will improve it by removing resources that are blocking the main thread of the JS engine in the browser. Also, it is crucial to remove unused JS and CSS files and perform lazy loading in images outside the browser viewport. With these operations we estimate to reduce this value in 1.08s and, thus, it is expected to reach the minimum threshold for the good category (<2000ms);
- TBT (60ms) - it is a replacement for the field-only FID metric. The current value is in the good category range (0 - 100ms). This value is obtained since in the implementation phase of LearnJS we take into account several tasks such as reducing JS execution time

and, consequently, minimizing its main-thread work. Also the reducing of third-party code by replacing large JavaScript libraries with smaller alternatives such as the JSHint and the infinite loop detection have a big impact in the final score;

- CLS (0.001) - the visual stability of LearnJs is very good (between 0 - 0.1) since we adhere to the good practices regarding this metric. One important factor was to flag the image dimensions in HTML in order to the browser reserve space prior to make the respective request. This way, later content is positioned in its final location affecting positively this metric.

4.2 Usability Evaluation

This section evaluates the usability of the graphical user interface of LearnJS based on the Nielsen's model [2], considered the most used usability heuristics for user interface design, analyzing factors such as usefulness and reliability. The usefulness factor relates the utility and usability offered by the system. The reliability is the ability of a system or component to perform its required functions under stated conditions for a specified period. The questionnaire was organized in 13 groups of questions, 12 regarding usefulness (visibility, compatibility, freedom, consistency, error prevention, recognition, flexibility, aesthetic, users help, help and documentation, ease of use and speed) and 1 for the reliability factor, with a total of 51 questions. Each group ended with a request for comments and the inquiry ends by asking to globally classify the LearnJS code playground. The questionnaire was distributed through an online survey, and disseminated to first- and second-year students of a Web Technologies degree at ESMAD, P.PORTO, which represent the target users of LearnJS. The respondents were informed that the questionnaire was anonymous, ensuring confidentiality. A total of 27 responses were gathered.



■ **Figure 2** Results of the usefulness (left) and reliability (right) heuristics.

The results showed that recognition, compatibility, and consistence are the aspects most valued by students, as more than 90% of students recognize these characteristics present always or almost always on the platform. Visibility, ease of use and documentation are also aspects highly valued by students. On the other hand, error prevention, speed and flexibility are the aspects, addressed in the questionnaire, less well evaluated by the students. The reliability section intended to check on what tasks students had difficulties, being the most reported ones the visualization of questions posed to the professor by other users, and overall and single exercise performance visualisations. Asked how they would classify the LearnJS playground, given all the parameters previously discussed, 96% of students rated

the platform as good or very good. Globally, some of the respondents' comments indicated the platform was very good, easy to use and learn, well structured and intuitive. On the other hand, some comments suggested the possibility of the platform to provide both basic and advanced solutions for the proposed problems.

5 Conclusion

This paper presents an user experience evaluation performed in the LearnJS code playground. The evaluation was focused on two important facets: performance and usability. The former was based on the lab and field data reported by the Google Lighthouse and PageSpeed Insights tools. The later was based on the user target perception collected in an online survey based on the Nielsen's heuristics.

Since LearnJS was designed keeping in mind these facets, the results obtained were more or less expected and demonstrates the importance of the awareness of these facets in the design phase of an interactive Web application. Nevertheless, the results pointed some issues that should be solved in the next version of the playground such as flexibility. In this topic students felt no liberty to customize the GUI (e.g. change panels location, alter the background of the editor), to schedule frequent actions (e.g. configure notifications) or the nonexistence of shortcut keys to perform the most used functions (e.g. run tests).




As future work, LearnJS will be capable to be integrated in an e-learning ecosystem based on an Learning Management System (e.g. Moodle, Sakai, BlackBoard). This integration will be based on the Learning Tools Interoperability (LTI) specification. This specification provides several interfaces for the authentication and grading services from/to the LMS (the tool consumer) and the LearnJS (the tool provider). Beyond integration, LearnJS will use in a near future a gamification engine to gamify the learning activities with level unblocking, rankings, experience points, badges and other gamification elements.

References

- 1 Toni Granollers. Usability evaluation with heuristics, beyond nielsen's list. In *ACHI 2018, The Eleventh International Conference on Advances in Computer-Human Interactions*, pages 60–65, 2018. URL: https://www.thinkmind.org/articles/achi_2018_4_10_20055.pdf.
- 2 J. Nielsen. *Usability engineering*. Academic Press, 1993. URL: <https://books.google.pt/books?id=fnvJ9PnbzJEC>.
- 3 Jacob Nielsen. 10 usability heuristics for user interface design, 1995. URL: <https://www.nngroup.com/articles/ten-usability-heuristics/>.
- 4 José Carlos Paiva, José Paulo Leal, and Ricardo Queirós. Game-Based Coding Challenges to Foster Programming Practice. In Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões, editors, *First International Computer Programming Education Conference (ICPEC 2020)*, volume 81 of *OpenAccess Series in Informatics (OASICS)*, pages 18:1–18:11, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/OASICS.ICPEC.2020.18.
- 5 José Carlos Paiva, José Paulo Leal, and Ricardo Queirós. Fostering programming practice through games. *Information*, 11(11), 2020. doi:10.3390/info11110498.
- 6 Ricardo Queirós. Learnjs - A javascript learning playground (short paper). In *7th Symposium on Languages, Applications and Technologies, SLATE 2018, June 21-22, 2018, Guimaraes, Portugal*, pages 2:1–2:9, 2018. doi:10.4230/OASICS.SLATE.2018.2.
- 7 Daniela Quiñones and Cristian Rusu. How to develop usability heuristics: A systematic literature review. *Computer Standards & Interfaces*, 53:89–122, 2017. doi:10.1016/j.csi.2017.03.009.

- 8 Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Longman Publishing Co., Inc., USA, 3rd edition, 1997.
- 9 B. Tognazzini. First principles, hci design, human computer interaction (hci), principles of hci design, usability testing. <http://www.asktog.com/basics/firstPrinciples.html>. Accessed: 2021-03-28.


Can I Code? User Experience of an Assessment Platform for Programming Assignments

Anne Münzner   

Center for Human-Computer Interaction, University of Salzburg, Austria

Nadja Bruckmoser 

University of Salzburg, Austria

Alexander Meschtscherjakov   

Center for Human-Computer Interaction, University of Salzburg, Austria

Abstract

Learning a programming language is a difficult matter with numerous obstacles for university students – but also for their lecturers. Assessment tools for programming assignments can support both groups in this process. They shall be adapted to the needs of beginners and inexperienced students, but also be helpful in long-term use. We utilised an adapted version of the Artemis system as an assessment platform for first-year computer science students in the introductory programming course. To examine the students' user experience (UX) over the semester, we conducted a three-stage online questionnaire study (N=42). We found that UX evolves over the semester and that platform requirements and problems in its usage change over time. Our results show that newcomers need to be addressed with caution in the first weeks of the semester to overcome hurdles. Challenges shall be added as the semester progresses.

2012 ACM Subject Classification Human-centered computing

Keywords and phrases Programming tool, user experience, student evaluation, programming assignment

Digital Object Identifier 10.4230/OASICS.ICPEEC.2021.18

Category Short Paper

Funding CodeAbility is funded by the Federal Ministry of Education, Science and Research as part of the structural funds for higher education.

1 Introduction

Learning to program can be a difficult task. Especially in the university context, when students are in the first semesters of their studies, they are not yet routinized and experienced with academic learning and problem solving and a completely new stage of life has often started with the study period [16, 7]. Students often have different levels of knowledge in programming. Some have already been taught solid basics in school, others have already experimented in the personal field, and still others have no previous experience at all.

But it is not easy for the lecturers either. They have to serve these different knowledge bases and are primarily occupied with creating and correcting suitable tasks, often for a large number of students. They have to create structures and incentives that make students do what is most important for learning a programming language: repeated intensive practice [13].

To address this problem, we have launched the CodeAbility project at Austrian universities which aims to standardize university programming education across Austria and create an infrastructure with an online learning platform, didactic concepts, and an exchange platform for teachers. The primary goal is to develop the online learning platform that integrates into the existing classroom using blended learning and, in the future, uses programming learning analytics to enable individual learning paths. We want to develop a platform that acts as a helping tool on the difficult path of learning to program and is also perceived as such.



© Anne Münzner, Nadja Bruckmoser, and Alexander Meschtscherjakov;
licensed under Creative Commons License CC-BY 4.0

Second International Computer Programming Education Conference (ICPEEC 2021).

Editors: Pedro Rangel Henriques, Filipe Portela, Ricardo Queirós, and Alberto Simões; Article No. 18;
pp. 18:1–18:12



OpenAccess Series in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Since frustration and fear are true learning killers [1], these must be avoided at all costs. Especially students who have never had contact with programming or computer science in general, have fears and doubts which create barriers that keep these people from learning. Especially for this group, it is important to convey a sense of self-efficacy, competence, security, and autonomy through the tool. This means not only avoiding frustration due to poor usability but specifically promoting positive experiences because lack of motivation is one of the main barriers to learning to program [1, 8].

At the moment, we are exploring what functionalities and content the platform should offer. Since the experience with delivery systems in higher education has been very positive [20] and support was needed most in this area, the functionality of the delivery platform was the first one we implemented and tested. In the first pilot phase of the project, Artemis, the automated assessment management system for interactive learning, was used in various courses at Austrian universities to create and submit exercises. Artemis was developed at the Technical University of Munich, Germany [10]. It is especially suited to supervise large groups and has been successfully used in MOOCs [11]. We have used the system at the University of Salzburg in the courses “Introduction to Programming with Java” and “Introduction to Programming with Python”. Both courses are aimed at students in the first bachelor’s semester of computer science and DIG (Digitization Innovation Society).

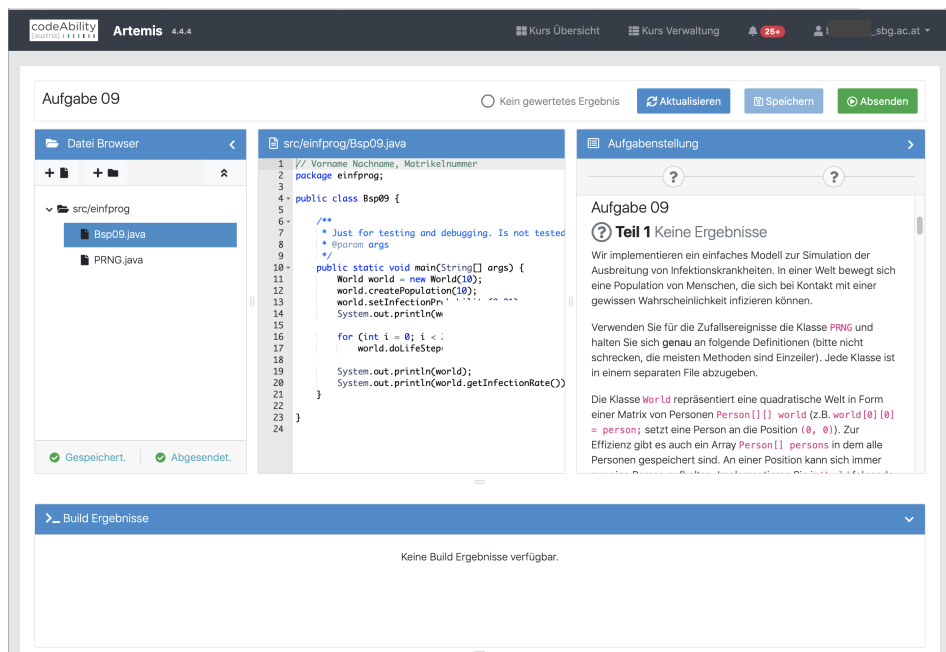
Figure 1 provides a screenshot of the Artemis platform as it was experienced by the students. An assignment was given that had to be submitted by a certain date. The interface includes a file browser (left), an area for the code that has to be developed (center), and the description of the assignment (right).

Today, there are many different assessment tools for programming assignments that save time and effort through automation [19] and have already been successfully used for programming courses at universities [2]. The requirements of lecturers and students in this context have also already been investigated [16].

As already stated, one of the problems concerns the motivation, i.e. also the emotional level of the students. We aim to develop a platform that also addresses this problem. Therefore, our main concern was to learn more about the students’ user experience (UX). It is already known that the UX of programs evolves over time with increasing experience [9]. For us, it was important to find out to what extent this also applies to our learning platform. As mentioned before, students are at a point in time where they are confronted with many new challenges, so it is particularly relevant to identify entry problems. But also the needs of students with growing experience should be recognised to optimally support different phases of learning programming. This leads to the following research questions:

- *How does the user experience of the learning platform changes during the first semester?*
- *How do requirements and reported problems change over the semester?*
- *How should deployed learning platforms be evaluated to capture the needs of all students?*

To answer these questions, we conducted a user study at the University of Salzburg with students who used the platform in their first semester. For this purpose, we sent out online questionnaires at three different times in the winter semester 20/21 (October 2020 - January 2021). Our study showed that the UX of the platform developed over the semester. We were given insights into existing problems with the platform that especially novice programmers found obstructive. Moreover, we were able to identify functions and aspects that students found particularly supportive. In the following, we will reflect on related work. We will describe the applied method in detail and report results. Lastly, we will discuss findings and provide answers to our research questions.



■ **Figure 1** This screenshot of the Artemis platform shows a file browser (on the left), a java code (in the center), and a student assignment (on the right). At the bottom information and errors would be reported. The green button at the top right allows the student to submit the code. An automated analysis (e.g., syntax errors, compilability) is performed and the student would get instant feedback at the bottom.

2 Related work

Solutions to the challenges of teaching programming are being explored in many countries through higher education institutions. Queirós et al. [16], for example, conducted a study in Portuguese universities to gain a deeper understanding of the programming teaching process in such higher education institutions. Their results show the need for helping tools that support teachers as well as students and automate parts of this process.

Cardoso et al. [2] conducted a study on the use of automated assessment tools for first semester students and demonstrated that the tools support and motivate students to learn programming.

The automated assessment system we used was Artemis, developed at the Technical University of Munich. In a quantitative study, its scalability, feedback under 10 seconds and good usability were proven [10]. The latter resulted from the observation that novice programmers are able to use the platform without problems. In addition, students gave positive feedback on the usefulness of the platform.

This proves that the use of automated assessment tools at universities is useful for teachers and students. For us, it was important to investigate not only the usability but also the UX in more detail in order to optimize the use of this tool, gain further insights into the needs of students over the semester, and thus gather knowledge for further development in the CodeAbility project.

There have been numerous studies on the change of UX and usability over time [15, 9]. Kujala et al. [12] describe a change in the context of an application over time under what they call long-term UX. It is the nature of learning to change learners through the growth of

knowledge and skills that build upon each other. It is a process with which the context, the target group of students changes. Since the tool accompanies the students in this process for months, the question of long-term UX is particularly relevant. When dealing with user experience in the context of learning, eudaemonic well-being is also important, as the enjoyment of the immediate activity is not always present or visible, but this activity can lead to increased well-being in the long run [5].

3 Methodology

3.1 Objective of the study

With our study, we wanted to gain deeper insights into the student perspective on the platform. Thus, the goal of the study was not simply to evaluate the existing platform, but to find out, with respect to the CodeAbility project, how students perceived the use of the platform, what is perceived as helpful, and where we need to change aspects. Since the platform is intended to support students as they get started in programming and to work for students who are less familiar with computer science, we evaluated the UX over the course of the semester. We were interested in whether the programming learning platform is rated differently over this time, for example, whether students rate it better with increasing experience or not. Thus, the overall objective of the study was the following: *Does the UX change over the semester and do students have different needs and issues with the platform at different points in time?*

3.2 Procedure

To evaluate UX, we used questionnaires that were sent to the students via the course instructors on three different dates during the semester. Typically, students had weekly classes and weekly assignments during the semester (i.e. over a four-month period). In the lectures, the students learned basic programming constructs such as data types, if-cases, and loops. A weekly task on the topic of the last lecture was then published via the Artemis platform. These assignments were divided into two subtasks that build on each other. Students could score 0 points if neither task passed the tests, 5 points if the first task passed the tests, and 10 points if both subtasks passed the tests. The students had to solve it in the course of the week. This task then had to be submitted via the Artemis platform either using the online editor or the version management system GIT. An instant check of the assignment was performed automatically and the student would get feedback from the Artemis platform. This feedback is based on the test cases created by the teachers. In addition to the compiler output, the students also received output on other tests, such as input output tests, and thus knew directly whether their solution passed all the tests. Students could resubmit their assignments as often as they wanted until the deadline. Thereafter, the solutions and the scores achieved were viewed by the lecturers and possibly assessed manually. On the platform, students could view their submitted assignments and achieved points.

The first survey was sent out six weeks after the start of the semester so that any initial difficulties were still fresh, but the students had already gained some experience with the platform. After another month a second questionnaire was sent out to students. The last questionnaire was sent out at the end of the semester after another six weeks.

We used the same questionnaire for all three dates. Demographic data from the participating students were collected in the first part of the questionnaire. The main purpose of this was to gain knowledge about the exact target group and find out whether it is a

relatively heterogeneous group or if there were differences among participants that need to be addressed. Here, we also asked questions about the students' previous experience in order to identify any correlations between this and the user experience of the platform.

The main part of the questionnaire was the *User Experience Questionnaire UEQ+*. The original UEQ was developed by Laugwitz et al. [14] and targeted at the assessment of UX of interactive systems in general. It measures the subjectively perceived impression of the users, mapped in six different dimensions: *Attractiveness*, *Efficiency*, *Transparency*, *Controllability*, *Stimulation*, and *Originality* [17]. The UEQ+ is a modular extension of the UEQ. It is composed of 26 bipolar questions rated using a 7-point Likert scale (coded from -3 to +3). The UEQ+ allows selecting various dimensions that are relevant for the respective product. We have ultimately used the scales *Attractiveness*, *Efficiency*, *Perspicuity*, *Dependability*, *Usefulness* and *Clarity* as UX dimensions.

The following part of the questionnaire included questions that further addressed the context of the application. These were questions about the platform's support for learning the programming language, the submission of assignments, and related error messages. These items were self-designed and had to be answered using a 10-level Likert scale from one (not at all) to 10 (very good).

The third part of the questionnaire consisted of open-ended questions about obstacles and difficulties as well as advantages, suggestions for improvement and wishes. They not only aimed to measure the user experience but also to provide possible justifications for the evaluation of this in the UEQ+ [18]. In addition, questions were intended to reveal further problems and potentials in the interaction at different points in the semester and provide the opportunity to obtain more qualitative answers [6].

3.3 Results

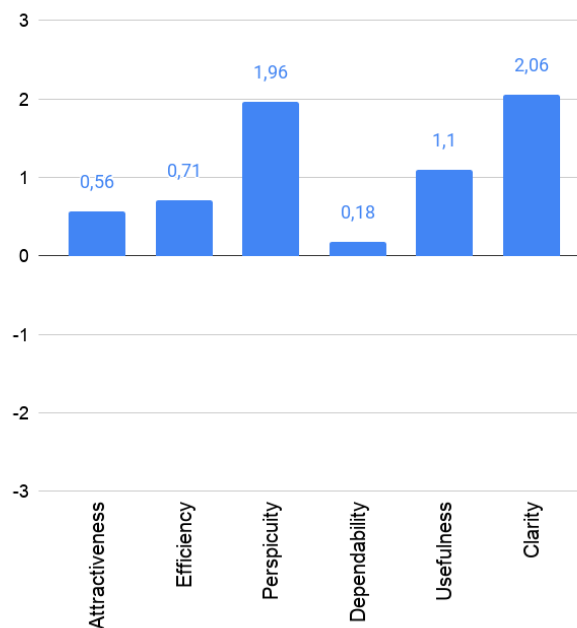
3.3.1 First Survey

A total of 61 students participated in the first survey. Of these, 42 students responded to the questionnaire in full. Since the introductory courses are intended for the first semester, 54.34% of the participants are in the first semester of their studies. The remaining participants are distributed among the other semesters.

12.8 % of the participants had no computer science classes in school and the majority with 31.9 % had 2 years of computer science classes in school. However, there are also some students (8.5 % of the participants) who have had computer science lessons in school for as long as eight years. This means that the previous school experiences were quite different. Figure 2 shows the results of the first UEQ+ questionnaire. Overall all six dimensions were rated positively (i.e. mean values above 0 on the scale between -3 and +3). *Dependability* scored worst with a value of 0.18. *Attractiveness* with 0.56 and *Efficiency* with 0.71 did not score very high either. *Usefulness* scored better with a value of 1.1. The best scores were *Perspicuity* with 1.96 and *Clarity* with 2.6. So far, we have not found any differences in the rating of the UEQ+ between students with different levels of prior knowledge.

The answers to the context-related questions were striking. They were answered very diversely. To the question "Artemis supports me in working on the homework", 14.3 % answered with *not at all* (0 points on the Likert scale) and the same number answered with *very well* (10 points on the Likert scale). The answer to the question "The error messages in the homework helped me" was similarly varied. Here, the proportion of those who voted neutral or worse (one to five points on the Likert scale) predominated.

18:6 Can I Code?



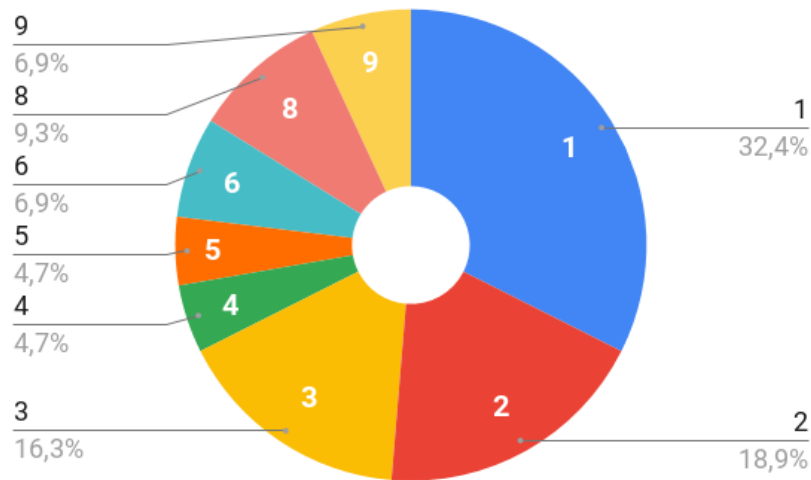
■ **Figure 2** Results of the UEQ+ of the first phase of the study. Ratings were given on a scale between -3 and +3. Mean values were all positive. *Dependability* scored worst. *Attractiveness* and *Efficiency* did not score very high. *Usefulness* scored better. The best scores were *Perspicuity* and *Clarity*.

However, the proportion of neutral to poor evaluations predominates in the question “Artemis has supported me in learning the programming language”. As shown in figure 3, 32.4 % of the participants stated that Artemis did not support them at all in this respect. Again, for all three questions, we have not yet found any differences in perceived support between students with different levels of prior knowledge.

In order to interpret these numbers, we shall have a look into the open questions. When asked what obstacles were encountered when working on and handing in the exercises, the majority of the participants criticized the quality of the error messages for the programmed exercises. These were described as inaccurate or incomprehensible. This problem is also reflected in the students’ wishes for improved and more detailed error messages that are also understandable for beginners.

Some students also found the submission via GIT difficult, especially for the first task, as they had to deal with the version management system in addition to the new algorithmic thinking. This problem is also reflected in the students’ requests for instruction for GIT.

But which aspects did the students rate positively? The immediate feedback of the platform on the submitted task was often praised as positive, even if the quality of the error messages was often criticized. A total of 24 of the 42 participants mentioned this as positive in the open questions. The students liked the fact that they could immediately see where their programs contained errors and correct them before the actual submission. In addition, they immediately had the certainty that their program corresponded to the created tests. So they did not have to wait for the lecturers to correct their solution.



■ **Figure 3** Results of the first study on the question: “Artemis has supported me in learning the programming language”. From 1 (not at all) to 10 (very good).

3.3.2 Development over time

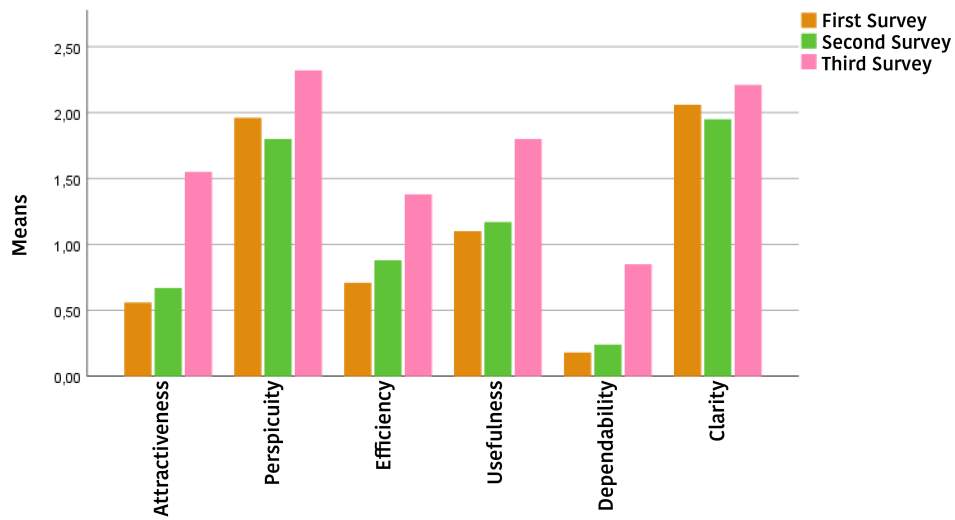
The comparison with the subsequent surveys showed that the UX of the platform increases in all the scales of the UEQ+ we examined as the semester progressed. In the second survey, the UEQ+ was still rated similarly to the first survey round. Either no or only minor improvements could be demonstrated. In the third round of the survey, greater improvements were detected. We saw a big increase in the dimensions *Attractiveness*, *Efficiency*, and *Usefulness*. Figure 4 shows detailed results of the UEQ+ over time.

The ratings of contextual questions also improved. In the last survey, for example, 11.5% did answer *not at all* to the question “Artemis supported me in learning the programming language”. For comparison: at the beginning of the study, 32.4% of the participants did so. The improvement can be seen in figure 5. The number of participants neglecting capability of the platform to support learning programming decreased significantly. Also, the support of the platform in the processing of the homework was rated better as the semester progressed as shown in figure 6. In the first survey participants were evenly spread across the answer possibilities (1 to 10). In the third survey, only a minority of participants stated that the platform would not be supportive for the assignments.

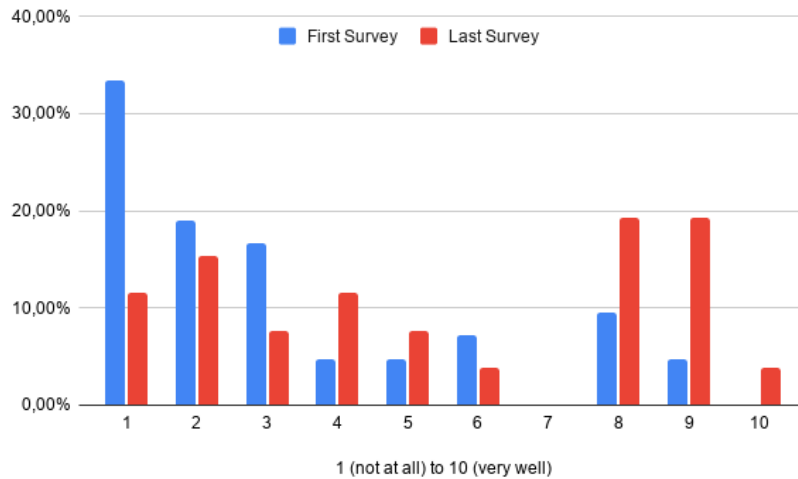
So are the problems identified in the first survey only the entry barriers described by Mendoza and Novick [15]? The answers to the open questions at the end of the questionnaire could give some clues. Now among the problems the lack of design freedom in the programming tasks, which does not allow own approaches since the system only accepts solutions that have been designed exactly according to the specifications, is mentioned much more often. However, experienced students would like to implement their own or more creative solutions. Even though the error outputs were still criticized, the complaints were more about the inaccuracy of these outputs.

The immediate feedback was also repeatedly emphasized as positive in these survey rounds. Furthermore, the students found it helpful to have all assignments in one place and thus to have an overview also of homework already handed in.

18:8 Can I Code?



■ **Figure 4** Results of the UEQ+ mean values over time. Mean values were taken from the three points in time of the study when the UEQ+ was asked. Especially the dimensions attractiveness, efficiency, usefulness, and dependability were rated higher at the end of the semester than at the beginning or during the semester.



■ **Figure 5** Detailed results on the statement "Artemis supported me in learning the programming language". Results show an increase in the capability of the platform as a support for programming language learning between the first and third survey.

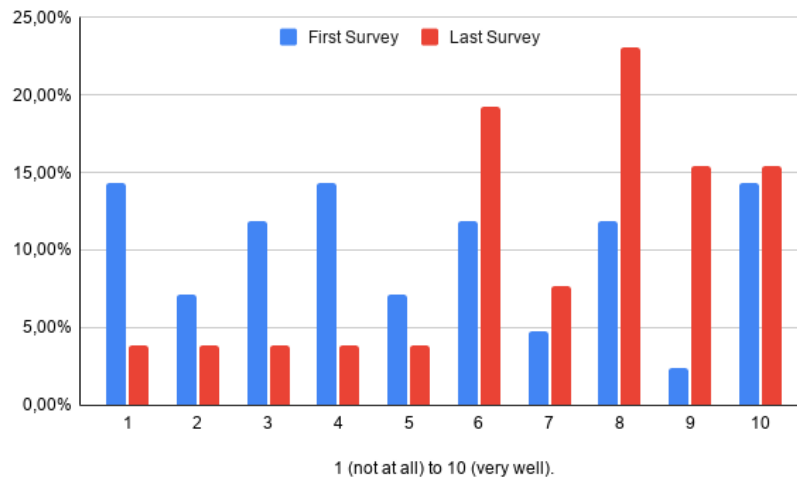


Figure 6 Detailed results on the statement “Artemis supported me in working on the assignments”. Results show an increase in the platform support for homework completion between the first and the third survey.

4 Discussion

In our study, we found that the user experience of using the Artemis platform at the beginning of the semester was lower compared to later points in the semester. In the beginning, the platform was not perceived as a support for learning to program by the majority of students. This was due to the fact that in addition to the new concepts of programming and algorithmic thinking that the students had to deal with at this point, they also had to learn and understand how to use and operate the Artemis platform. In addition to recognizing the workflow, completely new concepts and tools such as the version management system GIT and error output had to be understood.

On the one hand, this finding is gratifying, as it shows that the UX is perceived as more positive in the longer term and that the platform functions as a support for learning to program. Even though this development of the user experience has been proven over time in several studies, this finding is still very critical for us and no reason to rest on it, because as mentioned at the beginning, we are also concerned with simplifying the entry into programming.

4.1 Recommendations

If you want to analyze the user experience of learning platforms in higher education, our clear recommendation is to do this not only once and only at the end of the semester. Especially initial difficulties, which may have already led to leaving the course, can no longer be recorded here.

For many students, the online learning platform is their first point of contact with programming. The resulting responsibility and the potential to arouse students’ interest in programming and to create motivation for the rest of the semester should be used. Here the chance exists to get negative expectations and prejudices about programming [3, 8] out of the way.

Following the possibility-driven design approach [4], the potentials of an interactive learning platform should be used to create truly positive experiences here. According to our study, in addition to the clarity of the tasks and observation of one's progress, one of these potentials is the possibility of receiving immediate feedback. Isn't that also the beauty of programming and logic? Let's remember the moment when (maybe after a long time of debugging) the compiler no longer gives an error message and the algorithm does what it is supposed to do. Programming can be fun and this should be conveyed right at the beginning. Obstacles that stand in the way of this experience must be removed.

Of course, the platform must also be supportive for learning success in long-term use. This includes, for example, the use of GIT since it is a reality in later practical work on the job. In the same way, searching for errors and evaluating error messages that are often inaccurate or incomprehensible is part of a programmer's everyday work. These concepts must be learned, but should not overwhelm students all at once in their first programming assignment. They should be introduced gradually. For example, error messages could be filtered initially. In any case, the question of how feedback should be designed to support students at different levels of experience is worth exploring. In addition, as students become more experienced, the platform should also provide the opportunity for their own more creative approaches to solving the programming tasks, because the ability to find alternatives and to work on programming problems with different approaches is also important.

So it is obvious that a programming learning platform for students should function as a dynamic companion on the way of learning to program. With concepts like Programming Learning Analytics and Gamification, we want to achieve this. Whatever concepts we introduce in the future as part of the CodeAbility project, we will always pay special attention to what impact they have on the entry of still inexperienced students who have just started programming.

4.2 Limitations

In the pilot phase, the Artemis platform was used to create and submit weekly assignments and provide feedback on their correctness. We conducted our study only in the two courses that used the platform at the University of Salzburg (N=42). Thus, our results cannot be generalized to online learning platforms. This would require a further increase in the number of study participants. Furthermore, the platform used was limited to its function as a submission system for students. It remains to be investigated whether a learning platform with more extensive functionalities, such as teaching content, would be evaluated differently. Not to be neglected here is also the fact that the platform was used to submit weekly compulsory assignments, which influenced the grading of the students.

5 Conclusion

To support students and teachers at Austrian universities in learning and teaching programming, we started the CodeAbility project. In the first pilot phase of the project, we used the assessment tool Artemis to automate the creation and submission of assignments. In our study, we examined the user experience over the semester. According to our research questions, we found the following:

- *At the end of the semester, the user experience of the platform is rated better than at the beginning of the semester in all scales of the UEQ+, i.e. Attractiveness, Efficiency, Perspicuity, Dependability, Usefulness and Clarity. However, real differences in the evaluation could not yet be determined in the second, but only in the third survey at the end of the semester.*

- *At the beginning of the semester, in addition to the new computational thinking, students still struggle with initial difficulties in using the platform. Later in the semester, the problems with the platform and the students' demands change. Students no longer feel overwhelmed with new concepts and rather want more freedom in programming.*
- *To represent the user experience for the entire students, it makes sense to investigate of the user experience at least two different times, i.e. at the beginning and the end of the semester. This procedure captures both the entry problems of programming beginners and the needs of advanced students, which can differ considerably.*

In a future study, it would be helpful to conduct this survey with a larger number of students e.g. the other universities involved in the CodeAbility project.

On the one hand, our findings can be helpful for the evaluation of learning platforms, as it shows the requirement for repeated user experience studies. On the other hand, they can be useful for the design of learning platforms, as our findings show the need to address the different needs at the beginning and later in the semester. At the beginning of the semester, any obstacles that make it even more difficult to get started with programming should be eliminated. Then, as the semester progresses, additional concepts that students will need later should continue to be introduced. The platform should thus evolve along with the students.

References

- 1 Christine Bruce, Lawrence Buckingham, John Hynd, Camille McMahon, Mike Roggenkamp, and Ian Stoodley. The fear factor: How it affects students learning to program in a tertiary environment. *Journal of Information Technology Education: Research*, 9, January 2010. doi:10.28945/1183.
- 2 Marílio Cardoso, António Vieira de Castro, Álvaro Rocha, Emanuel Silva, and Jorge Mendonça. Use of Automatic Code Assessment Tools in the Programming Teaching Process. In Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões, editors, *First International Computer Programming Education Conference (ICPEC 2020)*, volume 81 of *OpenAccess Series in Informatics (OASICs)*, pages 4:1–4:10, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/OASICs.ICPEC.2020.4.
- 3 Sapna Cheryan, Allison Master, and Andrew Meltzoff. Cultural stereotypes as gatekeepers: Increasing girls' interest in computer science and engineering by diversifying stereotypes. *Frontiers in psychology*, 6:49, February 2015. doi:10.3389/fpsyg.2015.00049.
- 4 Pieter Desmet and Marc Hassenzahl. *Towards Happiness: Possibility-Driven Design*, pages 3–27. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. doi:10.1007/978-3-642-25691-2_1.
- 5 Sarah Diefenbach and Marc Hassenzahl. *Psychologie in der nutzerzentrierten Produktgestaltung*. Springer-Verlag Berlin Heidelberg, January 2017. doi:10.1007/978-3-662-53026-9.
- 6 Susan Farrell. Open-Ended vs. Closed-Ended Questions in User Research, 2016. URL: <https://www.nngroup.com/articles/open-ended-questions/>.
- 7 Anabela Gomes and António José Nunes Mendes. Learning to program-difficulties and solutions. *International Conference on Engineering Education*, 2007.
- 8 Tony Jenkins. ON THE DIFFICULTY OF LEARNING TO PROGRAM. In *3rd Annual LTSN-ICS Conference, Loughborough University*, 2002.
- 9 Evangelos Karapanos, Marc Hassenzahl, and Jean-Bernard Martens. User experience over time. In *Proceeding of the twenty-sixth annual CHI conference extended abstracts on Human factors in computing systems - CHI '08*, page 3561, New York, New York, USA, 2008. ACM Press. doi:10.1145/1358628.1358891.
- 10 Stephan Krusche and Andreas Seitz. ArTEMiS - An automatic assessment management system for interactive learning. In *SIGCSE 2018 - Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 2018. doi:10.1145/3159450.3159602.

- 11 Stephan Krusche and Andreas Seitz. Increasing the Interactivity in Software Engineering MOOCs - A Case Study. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019. doi:10.24251/hicss.2019.915.
- 12 Sari Kujala, Marlene Vogel, Anna E. Pohlmeyer, and Marianna Obrist. Lost in time: The meaning of temporal aspects in user experience. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, page 559–564, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2468356.2468455.
- 13 Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. A study of the difficulties of novice programmers. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITiCSE '05, page 14–18, New York, NY, USA, 2005. Association for Computing Machinery. doi:10.1145/1067445.1067453.
- 14 Bettina Laugwitz, Martin Schrepp, and Theo Held. Konstruktion eines Fragebogens zur Messung der User Experience von Softwareprodukten. In *Mensch und Computer 2006*, pages 125–134. OLDENBOURG WISSENSCHAFTSVERLAG, München, 2006. doi:10.1524/9783486841749.125.
- 15 Valerie Mendoza and David G. Novick. Usability over time. In *Proceedings of the 23rd annual international conference on Design of communication documenting & designing for pervasive information - SIGDOC '05*, page 151, New York, New York, USA, 2005. ACM Press. doi:10.1145/1085313.1085348.
- 16 Ricardo Queirós, Mário Pinto, and Teresa Terroso. Computer Programming Education in Portuguese Universities. *OpenAccess Series in Informatics*, 81(21):1–11, 2020. doi:10.4230/OASICS.ICPEC.2020.21.
- 17 Maria Rauschenberger, Martin Schrepp, and Jörg Thomaschewski. User experience mit fragebögen messen - durchführung und auswertung am beispiel des ueq. In *In Usability Professionals Konferenz 2013*, September 2013.
- 18 Heike Sandkühler, Martin Schrepp, and Jörg Thomaschewski. Ux messung mithilfe des ueq+ frameworks. In Christian Hansen, Andreas Nürnberger, and Bernhard Preim, editors, *Mensch und Computer 2020 - Workshopband*, Bonn, 2020. Gesellschaft für Informatik e.V. doi:10.18420/muc2020-ws105-244.
- 19 Draylson M. Souza, Katia R. Felizardo, and Ellen F. Barbosa. A Systematic Literature Review of Assessment Tools for Programming Assignments. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 147–156. IEEE, April 2016. doi:10.1109/CSEET.2016.48.
- 20 Anne Venables and Liz Haywood. Programming students need instant feedback! In *Proceedings of the Fifth Australasian Conference on Computing Education - Volume 20*, ACE '03, page 267–272, AUS, 2003. Australian Computer Society, Inc.