

# 4th International Computer Programming Education Conference

ICPEC 2023, June 26–28, 2023, Vila do Conde, Portugal

Edited by

Ricardo Alexandre Peixoto de Queirós  
Mário Paulo Teixeira Pinto



#### *Editors*

**Ricardo Alexandre Peixoto de Queirós** 

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

**Mário Paulo Teixeira Pinto** 

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

#### *ACM Classification 2012*

Applied computing → Interactive learning environments; Applied computing → E-learning; Applied computing → Computer-managed instruction; Applied computing → Computer-assisted instruction; Security and privacy

**ISBN 978-3-95977-290-7**

#### *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/978-3-95977-290-7>.

#### *Publication date*

August, 2023

#### *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.2023.0

ISBN 978-3-95977-290-7

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>**



## ■ Contents

Preface	
<i>Ricardo Alexandre Peixoto de Queirós and Mário Paulo Teixeira Pinto</i> .....	0:vii
List of Authors	
.....	0:ix
Committees	
.....	0:xi

### Invited Talk

Shifting Programming Education Assessment from Exercise Outputs Toward Deeper Comprehension	
<i>André L. Santos</i> .....	1:1–1:5

### Papers

I'm Sorry Dave, I'm Afraid I Can't Fix Your Code: On ChatGPT, CyberSecurity, and Secure Coding	
<i>Tiago Espinha Gasiba, Kaan Oguzhan, Ibrahim Kessba, Ulrike Lechner, and     Maria Pinto-Albuquerque</i> .....	2:1–2:12
LCSMAR, an AR Based Tool to Inspect Imperative Programs	
<i>Luis Carlos Martins, Lázaro Vinicius Lima, and Pedro Rangel Henriques</i> .....	3:1–3:9
Can a Content Management System Provide a Good User Experience to Teachers?	
<i>Yannik Bauer, José Paulo Leal, and Ricardo Queirós</i> .....	4:1–4:8
Sifu Reloaded: An Open-Source Gamified Web-Based CyberSecurity Awareness Platform	
<i>José Carlos Paiva, Ricardo Queirós, and Tiago Gasiba</i> .....	5:1–5:8
A Gamified Educational Escape Rooms' Framework for Computer Programming Classes	
<i>Ricardo Queirós, Carla Pinto, Mário Cruz, and Daniela Mascarenhas</i> .....	6:1–6:8
A Systematic Review of Formative Assessment to Support Students Learning Computer Programming	
<i>Jagadeeswaran Thangaraj, Monica Ward, and Fiona O'Riordan</i> .....	7:1–7:13
A New Approach to Perform Individual Assessments at Higher Education Using Gamification Systems	
<i>Filipe Portela</i> .....	8:1–8:12
NLP/AI Based Techniques for Programming Exercises Generation	
<i>Tiago Carvalho Freitas, Alvaro Costa Neto, Maria João Varanda Pereira,     and Pedro Rangel Henriques</i> .....	9:1–9:12



Data Visualization for Learning Analytics Indicators in Programming Teaching <i>Ranieri Alves dos Santos, Dalner Barbi, Vinicius Faria Culmant Ramos, and Fernando Alvaro Ostuni Gauthier</i> .....	10:1–10:7
Automated Assessment of Simple Web Applications <i>Luís Maia Costa, José Paulo Leal, and Ricardo Queirós</i> .....	11:1–11:8
Haskelite: A Step-By-Step Interpreter for Teaching Functional Programming <i>Pedro Vasconcelos</i> .....	12:1–12:8
A Systematic Review of Teacher-Facing Dashboards for Collaborative Learning Activities and Tools in Online Higher Education <i>Tiago Romão, Pedro Pestana, and Leonel Morgado</i> .....	13:1–13:12
An Experience with and Reflections on Live Coding with Active Learning <i>Anders Schlichtkrull</i> .....	14:1–14:13

## ■ Preface

As the world continues its rapid technological evolution, the role of computer programming education has become increasingly pivotal in shaping the future of our societies. The landscape of computer programming education is undergoing profound changes driven by advancements in technology, shifts in pedagogical approaches, and the growing demand for digital skills. As educators, researchers, and practitioners, we find ourselves confronted with new challenges and opportunities that call for a deep exploration of generative tools and their potential to revolutionize the teaching and learning of computer programming.

In this edition of ICPEC, we are excited to delve into a theme that not only reflects the current state of the field but also guides its future trajectory: “Generative Tools and the Future of Teaching-Learning in Computer Programming.”

Generative tools encompass a diverse array of technologies, from code generation frameworks to interactive development environments and automated assessment systems. These tools hold the promise of enhancing both the efficiency and effectiveness of programming education. With their ability to produce code, offer instant feedback, and facilitate collaborative learning, generative tools can empower students and educators alike to engage with programming concepts more deeply and creatively.

However, we must also confront a series of complex questions and challenges:

- How do generative tools impact students’ problem-solving skills and their understanding of fundamental programming concepts?
- What are the ethical considerations surrounding the use of automated assessment systems and code generation tools in educational contexts?
- How can educators best integrate generative tools into their curricula to promote a balanced approach between automation and conceptual mastery?
- What new paradigms of teaching and learning emerge as a result of incorporating generative tools, and how do these paradigms reshape the traditional classroom dynamics?

In addition to generative tools, this edition of ICPEC also highlights new trends, paradigms, and tools that are redefining the landscape of computer programming education. From the rise of interdisciplinary approaches that bridge programming with other fields, to the exploration of alternative programming paradigms and languages, we will explore the rich tapestry of possibilities that lie ahead.

We extend our deepest gratitude to the dedicated individuals who have contributed to making ICPEC 2023 a reality, from the organizing committee to the presenters and attendees. Your passion and commitment are the driving forces behind the success of this conference.

Ricardo Alexandre Peixoto de Queirós  
Mário Paulo Teixeira Pinto





## ■ List of Authors

Dalner Barbi  (10)  
Graduate Program in Knowledge Engineering  
and Management, UFSC, Florianópolis, Brazil

Yannik Bauer  (4)  
DCC – FCUP, Porto, Portugal;  
CRACS – INESC TEC, Porto, Portugal

Alvaro Costa Neto  (9)  
Instituto Federal de Educação, Ciência e  
Tecnologia de São Paulo, Barretos, Brazil

Luís Maia Costa  (11)  
Faculty of Engineering, University of Porto,  
Portugal;  
CRACS – INESC TEC, Porto, Portugal

Mário Cruz  (6)  
School of Education & inED, Polytechnic  
Institute of Porto, Portugal

Ranieri Alves dos Santos  (10)  
Graduate Program in Knowledge Engineering  
and Management, UFSC, Florianópolis, Brazil

Tiago Espinha Gasiba  (2)  
Siemens AG, München, Germany

Tiago Carvalho Freitas  (9)  
ALGORITMI Research Centre/LASI,  
University of Minho, Braga, Portugal

Tiago Gasiba  (5)  
Siemens AG, München, Germany;  
Universität der Bundeswehr München, Germany

Fernando Alvaro Ostuni Gauthier  (10)  
Graduate Program in Knowledge Engineering  
and Management, UFSC, Florianópolis, Brazil

Pedro Rangel Henriques  (3, 9)  
ALGORITMI Research Centre / LASI,  
DI-University of Minho, Braga, Portugal

Ibrahim Kessba  (2)  
Siemens AG, München, Germany

José Paulo Leal  (4, 11)  
CRACS – INESC TEC, Porto, Portugal;  
DCC – FCUP, Porto, Portugal

Ulrike Lechner  (2)  
Universität der Bundeswehr München, Germany

Lázaro Vinicius Lima  (3)  
Department of Informatics, Instituto Federal de  
Brasília, Brasil

Luis Carlos Martins  (3)  
ALGORITMI Research Centre / LASI,  
DI-University of Minho, Braga, Portugal

Daniela Mascarenhas  (6)  
School of Education & inED, Polytechnic  
Institute of Porto, Portugal

Leonel Morgado  (13)  
LE@D, CIAC, & CEG, Universidade Aberta,  
Lisbon, Portugal;  
INESC TEC, Porto, Portugal

Kaan Oguzhan  (2)  
Siemens AG, München, Germany

Fiona O’Riordan  (7)  
Teaching Enhancement Unit, Dublin City  
University, Ireland

José Carlos Paiva  (5)  
CRACS – INESC TEC, Porto, Portugal;  
DCC – FCUP, Porto, Portugal

Maria João Varanda Pereira  (9)  
Research Centre in Digitalization and Intelligent  
Robotics, Polytechnic Institute of Bragança,  
Portugal

Pedro Pestana  (13)  
Universidade Aberta, Lisbon, Portugal;  
CITAR, UCP, Porto, Portugal

Carla Pinto  (6)  
School of Engineering & CMUP, Polytechnic  
Institute of Porto, Portugal

Maria Pinto-Albuquerque  (2)  
Instituto Universitário de Lisboa (ISCTE-IUL),  
ISTAR, Portugal

Filipe Portela  (8)  
Algoritmi Centre, University of Minho,  
Guimarães, Portugal

Ricardo Queirós  (4, 5, 6, 11)  
CRACS – INESC TEC, Porto, Portugal;  
uniMAD – ESMAD, Polytechnic of Porto,  
Portugal

Vinicius Faria Culmant Ramos  (10)  
Graduate Program in Knowledge Engineering  
and Management, UFSC, Florianópolis, Brazil

Tiago Romão  (13)  
University of Trás-os-Montes e Alto Douro, Vila  
Real, Portugal;  
Universidade Aberta, Lisbon, Portugal;  
INESC TEC, Porto, Portugal

André L. Santos  (1)  
Instituto Universitário de Lisboa (ISCTE-IUL),  
ISTAR-IUL, Portugal

Anders Schlichtkrull  (14)  
Department of Computer Science, Aalborg  
University Copenhagen, Denmark

Jagadeeswaran Thangaraj  (7)  
School of Computing, Dublin City University,  
Ireland

Pedro Vasconcelos  (12)  
Departamento de Ciência de Computadores,  
Faculdade de Ciências da Universidade do Porto,  
Portugal;  
LIACC – Laboratório de Inteligência Artificial e  
Ciência de Computadores, Porto, Portugal

Monica Ward  (7)  
School of Computing, Dublin City University,  
Ireland

## ■ Committees

### Organizing Committee

Ricardo Queirós  
ESMAD, Polytechnic University of Porto  
Vila do Conde, Portugal  
ricardoqueiros@esmad.ipp.pt

Mário Pinto  
ESMAD, Polytechnic University of Porto  
Vila do Conde, Portugal  
mariopinto@esmad.ipp.pt

### Scientific Committee

Alberto Simões  
Portugal

Alexandre Braganca  
Portugal

Anabela Gomes  
Portugal

André Santos  
Portugal

Bárbara Cleto  
Portugal

Bertil P. Marques  
Portugal

Bruno Cunha  
Portugal

Cristina Alcaraz  
Spain

Daniela Pedrosa  
Portugal

Fabrizio Messina  
Italy

Filipe Pinto  
Portugal

Filipe Portela  
Portugal

Filomena Castro Lopes  
Portugal

Govardhan Aliseri  
India

Ivone Amorim  
Portugal

Jakub Swacha  
Poland

João Carlos Silva  
Portugal

João Cordeiro  
Portugal

J. Ángel Velázquez-Iturbide  
Spain

José Carlos Paiva  
Portugal

José Paulo Leal  
Portugal

Karolina Baras  
Portugal

Marco Temperini  
Italy

María Ángeles Pérez Juárez  
Spain

Maria José Marcelino  
Portugal

Mário Pinto, ESMAD  
Portugal

Martinha Piteira  
Portugal

Micaela Esteves  
Portugal

Muhammad Younas  
United Kingdom

Nuno Rodrigues  
Portugal

Patrícia Leite  
Portugal

**0:xii      Committees**

Paula Tavares  
Portugal

Pedro Rangel Henriques  
Portugal

Ricardo Martinho  
Portugal

Ricardo Queirós  
Portugal

Robertas Damasevicius  
Poland

Roberto Hirata Jr  
Brazil

Sergio Ilarri  
Spain

Simon Fong  
China

Sónia Sobral  
Portugal

Spyros Panagiotakis  
Greece

Štefan Korečko  
Slovak Republic

Teresa Terroso  
Portugal

Vítor Sá  
Portugal

Wei Zhou  
France

Ya-Han Hu  
China

# Shifting Programming Education Assessment from Exercise Outputs Toward Deeper Comprehension

André L. Santos  

Instituto Universitário de Lisboa (ISCTE-IUL), ISTAR-IUL, Portugal

---

## Abstract

Practice and assessment in introductory programming courses are typically centered on problems that require students to write code to produce specific outputs. While these exercises are necessary and useful for providing practice and mastering syntax, their solutions may not effectively measure the learners' real understanding of programming concepts. Misconceptions and knowledge gaps may be hidden under an exercise solution with correct outputs. Furthermore, obtaining answers has never been so easy in the present era of chatbots, so why should we care (much) about the solutions? Learning a skill is a process that requires iteration and failing, where feedback is of utmost importance. A programming exercise is a means to build up reasoning capabilities and strategic knowledge, not an end in itself. It is the process that matters most, not the exercise solution. Assessing if the learning process was effective requires much more than checking outputs.

I advocate that introductory programming learning could benefit from placing more emphasis on assessing learner comprehension, over checking outputs. Does this mean that we should not check if the results are correct? Certainly not, but a significant part of the learning process would focus on assessing and providing feedback regarding the comprehension of the written code and underlying concepts. Automated assessment systems would reflect this shift by comprising evaluation items for such a purpose, with adequate feedback. Achieving this involves numerous challenges and innovative technical approaches. In this talk, I present an overview of past and future work on tools that integrate code comprehension aspects in the process of solving programming exercises.

**2012 ACM Subject Classification** Social and professional topics → Computer science education; Applied computing → Computer-assisted instruction

**Keywords and phrases** Introductory programming, assessment, comprehension

**Digital Object Identifier** 10.4230/OASISs.ICPEC.2023.1

**Category** Invited Talk

**Acknowledgements** I thank the ICPEC organizing committee for this Invited Talk.

## 1 Do programming learners fully understand their code?

Studies have shown that programming assignments that are successfully solved do not necessarily have a matching learner confidence [13, 11], while a significant number of students may struggle to explain their own code [15]. In other words, a learner's ability to write a correct solution to a problem does not imply mastery of the underlying concepts, algorithms, and programming primitives. Despite reaching solutions that work, the learner may hold misconceptions [7, 18] about the written code. Even if a learner did not cheat and actually wrote the code, the latter could have been obtained through tinkering and trial-and-error until reaching a working solution. In my experience as a programming instructor, I often get surprised when a third-year student almost graduating cannot interpret rather elementary aspects of program execution and errors. (I ask myself: *How did the student reach this point without understanding these matters? Systematically hammering out programs until they work as expected?*)



© André L. Santos;  
licensed under Creative Commons License CC-BY 4.0

4th International Computer Programming Education Conference (ICPEC 2023).

Editors: Ricardo Alexandre Peixoto de Queirós and Mário Paulo Teixeira Pinto; Article No. 1; pp. 1:1–1:5

OpenAccess Series in Informatics



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

## 1:2 Shifting Programming Education from Exercise Outputs Toward Comprehension

These days we have just reached the era of chatbots capable of correctly solving most introductory programming assignments [8]. These systems, such as the popular ChatGPT<sup>1</sup>, not only solve problems but also have a generative nature that allows them to output different solutions for those, with accompanying detailed explanations. Until now one could easily find code snippets for typical programming problems by browsing the Web. However, the learner had to make at least a minimal effort regarding the interpretation and integration of the search. With the advent of the latest chatbots, that effort is reduced to the minimum possible – copy and paste the problem statement. Furthermore, there are barely any constraints regarding the written or programming language. This implies that one may easily pass an online programming exam blindly, just using the outputs of chatbots. Some solutions will fail or not be optimal, but the overall score is likely to be positive. Therefore, obtaining solutions has never been so easy, and this is likely to remain as such.

If just reading solutions would be adequate to learn to program, chatbots would not even be necessary, and students simply would be provided with a bundle of problem statements and corresponding solutions.<sup>2</sup> Most programming exercises are “classical”, for which thousands of solutions exist out there. The ease of obtaining solutions implies that they lose their value. However, the real benefit of solving a programming exercise is not reaching a solution, but rather the process of developing it – the problem interpretation, the strategy to approach it, how to express it, handling common errors and bugs that will occur, and in turn, their interpretation and strategy to overcome those. If a learner obtains immediate working solutions without going through this process and just focuses on checking whether they meet the desired outcome, will certainly acquire weaker skills. In analogy with natural languages, it is like training to write sentences by modifying existing ones, but not being able to express oneself by writing on a blank sheet.

Even in a pre-chatbot era, overreliance on automatic assessment systems could lead to an “autograder insanity” phenomenon [5], where students replace the task of testing their own solutions with highly frequent resubmissions to the system until they obtain a high score. Not imposing limits and/or penalties for overusing the submission system may incentivize a trial-and-error and tinkering approach to problem-solving. I believe this is a poor educational practice, because one may reach a solution with high quality, but not fully understand all aspects that characterize it as such. While in a small programming exercise against an autograder a trial-and-error tactic may suffice to reach a working solution, in real settings that strategy may be highly inefficient due to aspects pertaining to state space size, system complexity, and development settings.

Access to (reliable) information is generally perceived as beneficial. Still, I argue that the absence of learning strategies that are adapted to this new reality may hinder programming learning processes (and other subjects). Struggling students will likely use the means at their disposal to overcome their difficulties, especially when they are considered legitimate. I believe that if the activities for fulfilling programming course requirements remain focused on obtaining solutions, the widespread use of chatbots might lead to a shallow acquisition of programming skills.

---

<sup>1</sup> <https://chat.openai.com/>

<sup>2</sup> Worked examples [4] are an effective learning means, but their aim is not to replace deliberate practice.

## 2 How can courseware help to improve program comprehension?

I believe that the instructional design of programming courses could place more emphasis on assessing the learner's comprehension of programming concepts, algorithms, and code understanding while downplaying the accomplishment of reaching a solution that produces the expected outputs. In this perspective, an exercise would not be completed until some assessment of program comprehension is carried out. My hypothesis is that deemphasizing solution outputs will, to some degree, shift the learners' attention and learning time to understanding.

A possible approach to assess code understanding is to pose questions about learners' code [16]. Given that having a human tutor to carry out this role systematically for each individual learner is likely not to be feasible in practice due to instructor availability and cost, such an approach would better scale using automated assessment systems. A recent survey [17] concluded that such systems generally do not comprise this sort of meta-cognition feedback on the submitted code solutions. Jask [21] is a research prototype capable of generating question-answer pairs about Java code against methods. Our early experiment with introductory programming students revealed a high failure rate ( $> 60\%$ ) on questions involving program dynamics (e.g., variable tracing, call stack) [21], while their solutions were producing the expected outputs. Another study with questions on JavaScript [14] has revealed similar failure rates and found that students that repeatedly fail these questions are more likely to drop out. The results confirm that correct exercise solutions do not imply an understanding of the inner workings of programs. This fragility may become evident as problems and algorithms get more complex, but that was not yet evaluated.

Questions about learners' code could be applied in a post-submission fashion in an automated assessment system, where a learner would face questions about the submitted code solution. Upon submitting a solution, the system could also ask questions about other solutions for the same exercise from other students. This would lead the learner to carry out code comprehension tasks related to the same matter, possibly strengthening the related skills by having to interpret similar or different solutions.

Questions about learners' code may be posed at different moments and target different concerns. For example, an inquisitive code editor [10] may prompt questions to learners when the written code reveals a hypothetical misconception, fostering users to reflect on their code. Another possibility for asking questions is during debugging (when errors occur) [1], leading users to reflect on the cause of the errors (instead of trying something else straight away).

Another form of assisting programming learners during exercise solving is by providing feedback and appropriate hints. When a learner cannot progress in an exercise the first temptation might be to look for the solution somewhere else, such as a chatbot. However, in their current form, chatbots may straightly provide a complete solution. As discussed earlier, this may cause the learner to go through the exercise with minimal reflection, despite that chatbots are also capable of outputting detailed explanations of the provided solutions. Jinter [9] is a system to provide fine-grained hints for progressing and receiving feedback in programming exercises. Instead of providing straight answers, the hints attempt at leading the learner to a viable path, while demanding some reflection. Other approaches have focused on providing feedback for improving the code, for instance, with respect to code quality [2] and refactoring [12].

Following a different research line – educational programming environments (e.g., Ville [19], BlueJ [6], Thonny [3], PandionJ [20]) – I postulate that the programming environments themselves could improve their role in program comprehension. Namely, by providing

additional insights that would be otherwise unnoticeable or of difficult access, such as providing facilities for users to: see execution history, ask questions about program behavior, trace output to program statements, detail error explanations and location, and present information about execution performance (time and memory). Hypothetically, the absence of available information to help understand what went wrong in a program incentivizes a learner to search for other forms of overcoming the problem.

To conclude, chatbots have the potential of being a fabulous aid to programming learners that are stuck in their progress. The great novelty of chatbots relates to their immediacy and conversational nature, while they do not provide anything that is not explained elsewhere. Programming education will have to adapt to this new reality, but I argue that educators should not overvalue chatbots as if they were a silver bullet – in the end, one still has to understand how programs work. Time will tell how chatbots affect the learning processes of programming.

---

### References

- 1 Fatima Abu Deeb and Timothy Hickey. Reflective debugging in Spinoza V3.0. In *Australasian Computing Education Conference, ACE '21*, pages 125–130, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3441636.3442313.
- 2 Francisco Alfredo, André L. Santos, and Nuno Garrido. Sprinter: A didactic linter for structured programming. In Alberto Simões and João Carlos Silva, editors, *Third International Computer Programming Education Conference, ICPEC 2022, June 2-3, 2022, Polytechnic Institute of Cávado and Ave (IPCA), Barcelos, Portugal*, volume 102 of *OASICs*, pages 2:1–2:8. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/OASICs.ICPEC.2022.2.
- 3 Aivar Annamaa. Introducing Thonny, a Python IDE for learning programming. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, Koli Calling '15, pages 117–121, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2828959.2828969.
- 4 Robert K. Atkinson, Sharon J. Derry, Alexander Renkl, and Donald Wortham. Learning from examples: Instructional principles from the worked examples research. *Review of Educational Research*, 70(2):181–214, 2000. doi:10.3102/00346543070002181.
- 5 Elisa Baniassad, Lucas Zamprogno, Braxton Hall, and Reid Holmes. Stop the (autograder) insanity: Regression penalties to deter autograder overreliance. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, SIGCSE '21*, pages 1062–1068, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3408877.3432430.
- 6 Jens Bennedsen and Carsten Schulte. BlueJ visual debugger for learning the execution of object-oriented programs? *ACM Transactions on Computing Education*, 10(2):8:1–8:22, June 2010. doi:10.1145/1789934.1789938.
- 7 Luca Chiodini, Igor Moreno Santos, Andrea Gallidabino, Anya Taffiovich, André L. Santos, and Matthias Hauswirth. A curated inventory of programming language misconceptions. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1, ITiCSE '21*, pages 380–386, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3430665.3456343.
- 8 James Finnie-Ansley, Paul Denny, Brett A. Becker, Andrew Luxton-Reilly, and James Prather. The robots are coming: Exploring the implications of openai codex on introductory programming. In *Proceedings of the 24th Australasian Computing Education Conference, ACE '22*, pages 10–19, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3511861.3511863.
- 9 Jorge Gonçalves and André L. Santos. Jinter: a hint generation system for Java exercises. In *28th annual ACM conference on Innovation and Technology in Computer Science Education (ITiCSE) (to appear)*, 2023.

- 10 Austin Z. Henley, Julian Ball, Benjamin Klein, Aiden Rutter, and Dylan Lee. An inquisitive code editor for addressing novice programmers' misconceptions of program behavior. In *43rd IEEE/ACM International Conference on Software Engineering: Software Engineering Education and Training, ICSE (SEET) 2021, Madrid, Spain, May 25-28, 2021*, pages 165–170. IEEE, 2021. doi:10.1109/ICSE-SEET52601.2021.00026.
- 11 Cazembe Kennedy and Eileen T. Kraemer. Qualitative observations of student reasoning. In *The 24th Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE '19*, pages 224–230. ACM, 2019. doi:10.1145/3304221.3319751.
- 12 Hieke Keuning, Bastiaan Heeren, and Johan Jeuring. A tutoring system to learn code refactoring. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, SIGCSE '21*, pages 562–568, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3408877.3432526.
- 13 Päivi Kinnunen and Beth Simon. My program is ok – am I? computing freshmen's experiences of doing programming assignments. *Computer Science Education*, 22(1):1–28, 2012. doi:10.1080/08993408.2012.655091.
- 14 Teemu Lehtinen, Lassi Haaranen, and Juho Leinonen. Automated questionnaires about students' JavaScript programs: Towards gauging novice programming processes. In *Proceedings of the 25th Australasian Computing Education Conference, ACE 2023, Melbourne, VIC, Australia, 30 January 2023 - 3 February 2023*, pages 49–58. ACM, 2023. doi:10.1145/3576123.3576129.
- 15 Teemu Lehtinen, Aleksi Lukkarinen, and Lassi Haaranen. Students struggle to explain their own program code. In Carsten Schulte, Brett A. Becker, Monica Divitini, and Erik Barendsen, editors, *ITiCSE '21: Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V.1, Virtual Event, Germany, June 26 - July 1, 2021*, pages 206–212. ACM, 2021. doi:10.1145/3430665.3456322.
- 16 Teemu Lehtinen, André L. Santos, and Juha Sorva. Let's ask students about their programs, automatically. In *29th IEEE/ACM International Conference on Program Comprehension, ICPC 2021, Madrid, Spain, May 20-21, 2021*, pages 467–475. IEEE, 2021. doi:10.1109/ICPC52881.2021.00054.
- 17 José Carlos Paiva, José Paulo Leal, and Álvaro Figueira. Automated assessment in computer science education: A state-of-the-art review. *ACM Trans. Comput. Educ.*, 22(3), June 2022. doi:10.1145/3513140.
- 18 Yizhou Qian and James Lehman. Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Trans. Comput. Educ.*, 18(1), October 2017. doi:10.1145/3077618.
- 19 Teemu Rajala, Mikko-Jussi Laakso, Erkki Kaila, and Tapio Salakoski. Ville: A language-independent program visualization tool. In *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research - Volume 88, Koli Calling '07*, pages 151–159, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc. URL: <http://dl.acm.org/citation.cfm?id=2449323.2449340>.
- 20 André L. Santos. Enhancing visualizations in pedagogical debuggers by leveraging on code analysis. In Mike Joy and Petri Ihantola, editors, *Proceedings of the 18th Koli Calling International Conference on Computing Education Research, Koli, Finland, November 22-25, 2018*, pages 11:1–11:9. ACM, 2018. doi:10.1145/3279720.3279732.
- 21 André L. Santos, Tiago Soares, Nuno Garrido, and Teemu Lehtinen. Jask: Generation of questions about learners' code in Java. In Brett A. Becker, Keith Quille, Mikko-Jussi Laakso, Erik Barendsen, and Simon, editors, *ITiCSE 2022: Innovation and Technology in Computer Science Education, Dublin, Ireland, July 8 - 13, 2022, Volume 1*, pages 117–123. ACM, 2022. doi:10.1145/3502718.3524761.



# I'm Sorry Dave, I'm Afraid I Can't Fix Your Code: On ChatGPT, CyberSecurity, and Secure Coding

**Tiago Espinha Gasiba** ✉   
Siemens AG, München, Germany

**Kaan Oguzhan** ✉   
Siemens AG, München, Germany

**Ibrahim Kessba** ✉   
Siemens AG, München, Germany

**Ulrike Lechner** ✉   
Universität der Bundeswehr München, Germany

**Maria Pinto-Albuquerque** ✉   
Instituto Universitário de Lisboa (ISCTE-IUL), ISTAR, Portugal

---

## Abstract

Software security is an important topic that is gaining more and more attention due to the rising number of publicly known cybersecurity incidents. Previous research has shown that one way to address software security is by means of a serious game, the CyberSecurity Challenges, which are designed to raise awareness of software developers of secure coding guidelines. This game, which has been proven to be very successful in the industry, makes use of an artificial intelligence technique (laddering technique) to implement a chatbot for human-machine interaction.

Recent advances in machine learning led to a breakthrough, with the implementation of ChatGPT by OpenAI. This algorithm has been trained in a large amount of data and is capable of analysing and interpreting not only natural language, but also small code snippets containing source code in different programming languages. With the advent of ChatGPT, and previous state-of-the-art research in secure software development, a natural question arises: *to which extent can ChatGPT aid software developers in writing secure software?*

In this paper, we draw on our experience in the industry, and also on extensive previous work to analyse and reflect on how to use ChatGPT to aid secure software development. Towards this, we run a small experiment using five different vulnerable code snippets. Our interactions with ChatGPT allow us to conclude on advantages, disadvantages and limitations of the usage of this new technology.

**2012 ACM Subject Classification** Applied computing → Learning management systems; Security and privacy → Software security engineering; Applied computing → Distance learning; Applied computing → E-learning

**Keywords and phrases** Serious Games, IT-Security, Machine Learning, ChatGPT, Secure Coding, Industry, Software Development, Teaching

**Digital Object Identifier** 10.4230/OASICS.ICPEC.2023.2

**Funding** *Ulrike Lechner*: Acknowledged funding for project LIONS by dtec.bw.

*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, Maria Pinto-Albuquerque acknowledges and thanks the Instituto Universitário de Lisboa and ISTAR, for their support.



© Tiago Espinha Gasiba, Kaan Oguzhan, Ibrahim Kessba, Ulrike Lechner, and Maria Pinto-Albuquerque;  
licensed under Creative Commons License CC-BY 4.0

4th International Computer Programming Education Conference (ICPEC 2023).

Editors: Ricardo Alexandre Peixoto de Queirós and Mário Paulo Teixeira Pinto; Article No. 2; pp. 2:1–2:12



OpenAccess Series in Informatics

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

## 1 Introduction

According to ISO 25000 [16], one aspect of software development is security. Software security has been gaining much attention over the last decade due to the increasing number of cybersecurity incidents that are caused by poor software development practices. As a consequence, industrial standards such as IEC 62.443 mandate the implementation of a secure software development life cycle, to address and lower the number of vulnerabilities in products and services. The development of secure software is not only an important topic for the industry (e.g. in critical infrastructures), but it is also an important subject taught in many engineering and informatics courses at several universities.

There are several known methods to improve the quality of software. Among others, some of these methods include: performing secure code reviews, usage of static application security testing (SAST), and employment of security testing techniques such as unit testing, penetration testing, and fuzzing. These methods to improve software security are generally based on the fact that software should comply to a set of secure coding guidelines; secure coding guidelines are policies aimed at minimizing vulnerabilities and bugs in software. One way to ensure that software follows secure coding guidelines is by means of the usage of static application security testing tools. However, not all secure coding guidelines are decidable [3], i.e. there exist some secure coding guidelines for which no theoretical Turing Machine (TM) can be constructed such that, given some source code the TM identifies compliance or non-compliance to the guideline. As a result from this theoretical perspective, an immediate problem is raised: full automation of secure coding is not possible. Software developers are ultimately responsible for the security of the code they write. However, in a 2019 survey with more than 4000 software developers from the industry, Patel [21] has shown that more than 50% of them cannot recognize vulnerabilities in source code.

A way to address this problem is to raise awareness of secure coding among software developers. Similar to Patel, in [11], Gasiba has shown that industrial software developers' lack awareness of secure coding guidelines. He extended the work by Hänsch et al. to the field of secure coding, defining secure coding awareness in three dimensions: perception, protection, and behavior.

The recent advances in technology, in particular in Machine Learning (ML), allow new techniques to be used to assist software developers to write secure code. In [8], Gasiba et al. have shown that artificial intelligence can be used to raise awareness of software developers. The authors devised an intelligent coach by means of an artificial intelligent technique - the laddering technique - which is mostly used in chatbots [22]. The intelligent coach, which allowed Human-Machine interaction (HMi) in a controlled environment (the Sifu platform), was shown to be very successful to raise awareness of secure coding guidelines of software developers in the industry.

In this paper, we extend previous work by exploring the usage of ChatGPT [19] as a means of HMi. ChatGPT, which was released in November 2022, is built on top of the GPT-3 family of large language models, and was developed by the American research laboratory OpenAI. The language model has been fine-tuned with both supervised and reinforcement learning techniques.

Given the authors' experience, previous work, as also the theoretical limitations inherent to the secure coding field, this work aims to broaden the understanding on the extent ChatGPT can aid software developers to write secure code. This work seeks to understand *to which extent ChatGPT can recognize vulnerabilities in source code*, and *to which extent ChatGPT can rewrite code to eliminate the present security vulnerabilities*. The reason the

authors have chosen ChatGPT for experimentation, as opposed to other existing generative models (in particular models trained for cybersecurity), is the fact that, to the best of our knowledge, not only is ChatGPT available to the wide public, but also allows to maintain a conversation while remembering previous requests and answers. Therefore, we achieve the present study by means of interactions with ChatGPT based on five exercises taken from the serious game CyberSecurity Challenges (CSC), and analysis of the answers in terms of secure coding. In the present work, we also reflect on the usage of ChatGPT and similar technologies as a means to teach software developers to write secure code, both in academia and also in the industry.

This work provides a valuable insight to both industry practitioners, but also to researchers, by giving an overview of the advantages, disadvantages but also limitations of using Human-Machine interactions as a means to raise awareness of secure coding. This paper also opens the doors and gives a first step towards a new and rich field of research: using Machine Learning algorithms and generative AI to raise awareness of secure coding by means of HMI.

In Section 2 we present related work that was used as basis for our research. In Section 3 we describe the setup and experiment that we used to address our research question. Section 4 presents the result as the outcome of our experiments. A discussion and reflection on the obtained results is provided in Section 5. Finally, Section 6 concludes the paper and gives an outline of future work.

## 2 Related Work

Several previous work was used as a basis for the current publication. Industrial security standards such as ISO/IEC 62.443 [15] motivate the work. In particular, the 4.1 part of the standard describes the implementation of processes to address the life cycle of secure software development. One important aspect that the standard mandates is the establishment of several secure coding practices, e.g. the implementation of secure coding guidelines during the development of software. Some influential secure coding guidelines are provided by the Open Web Application Security Project (OWASP) in the form of Top-10 rules [18], and the secure coding guidelines provided by the Software Engineering Institute of the Carnegie Mellon University [5]. A further cybersecurity standard widely used in the industry is given by the MITRE corporation in form of Common Weakness Enumeration [6].

One way to address IT security is given by the German BSI Grundschriftkatalog [4] standard, which recognizes serious games as a means to raise awareness of IT security. A serious game, as defined by Dörner et al. [7], is a game that is developed with a purpose that is not only entertainment. Gasiba et al. have developed a game with the purpose to raise awareness of software developers of secure coding guidelines. This game (the CyberSecurity Challenges) is based on a platform which the authors called Sifu [8, 11], and has been shown to motivate software developers to think about security. In their game, the player is presented with a secure coding challenge containing software vulnerabilities. The player interacts with an intelligent coach, which is a software component that implements an artificial intelligence (AI) engine, in order to solve the challenge. The goal of the intelligent coach is to provide hints to the players on the reasons why software is not compliant to secure coding rules. The game is played through several interactions with the intelligent coach, until a solution to the given challenge is considered acceptable by the AI algorithm. The criteria for an acceptable solution includes: (1) initial vulnerability present in the challenge is removed, (2) no additional vulnerabilities are introduced, and (3) the code respects the desired semantics, i.e. behaves as expected. The AI mechanism implemented in the Sifu platform makes use of the chatbot laddering technique [22]. The hints that are provided to the player are thus given in an increasing level of clarity and exactness.

Although there exist other ways to increase software security, e.g. by means of static application security testing tools, our work focuses on the human factor. We motivate our choice by the fact that secure coding is a topic that cannot be fully solved by means of automation [13, 20, 2]. In [1], Acar et al. discuss how software developers search for advice for their coding activities, and conclude that software developers need assistance to understand this advice. Additionally, in 2019, Patel [21] conducted a large-scale study with over 4000 software developers. One of the results of his survey shows that more than half of software developers cannot recognize vulnerabilities in source code. Similar results were obtained by Gasiba et al. [9, 10].

Recent work has been published on the usage of machine learning algorithm to detect security vulnerabilities in source code. Harrer et al. [14] studied two feature extraction methods for C/C++ code and use this to build a control flow graph and to determine if the code contains vulnerabilities or not, without classifying them. In [23], Tang et al. extend previous work by not only looking for the presence of vulnerabilities, but also in classifying them according to MITRE's Common Weakness Enumeration (CWE) [6]. While previous approaches dealt with the C and C++ programming languages, Louati et al. extended this work for the C# programming language in [17]. To the best of our knowledge, all previous work show good indicators that machine learning is adequate to detect and classify vulnerabilities in software.

One machine learning algorithm that is currently raising lots of interest in the research community is the ChatGPT. ChatGPT is a language model developed by OpenAI that uses machine learning to generate human-like text. It is trained on a large dataset of text from the internet and is capable of understanding and responding to natural language inputs. It can be used for a wide range of tasks such as language translation, question answering, and text completion.

Not only is the algorithm based on machine learning, but its implementation allows a natural dialog between man and machine. ChatGPT processes queries from users, which are written in English, and compute an answer in an conversational way. ChatGPT not only processes the answer based on the current query, but also based on previous queries. Although it is mostly trained for natural languages, it can also interpret programming languages, such as C and C++. In the present work, we use this feature to conduct an interactive dialog with the algorithm based on five challenges from the CSC game.

### 3 Experiment

To setup our experiment, we selected five different challenges from the CyberSecurity Challenges, based on C/C++, and contained in the Sifu platform. Table 1 shows a summary of the selected challenges, and the corresponding CWE identifier. These challenges were chosen based both on the prevalence of the programming errors, but also based on practical experience in teaching cybersecurity from the authors'. The authors' used the 2023 January 13 version of ChatGPT, which is based on the GPT-3 training data.

The first challenge contains code of a C function that has a standard buffer overflow vulnerability. The buffer overflow in this challenge is evident through the usage of the *strcpy* function. The function in the second challenge, which is developed in C++, contains vulnerability based on undefined behavior. Depending on the compiler, the implemented function can produce different results. The third challenge makes use of a vulnerable C function, the *gets* function. Due to the problems that this function can cause, it has been deprecated and removed in the C11 standard. The fourth challenge corresponds to code that

■ **Table 1** Selected Challenges from Sifu Platform, According to CWE ID.

ID	Vulnerability	Description
1	CWE-121	Stack-Based Buffer Overflow
2	CWE-758	Reliance on Undefined, Unspecified, or Implementation-Defined Behavior
3	CWE-242	Use of Inherently Dangerous Function
4	CWE-190	Integer Overflow or Wraparound
5	CWE-208	Observable Timing Discrepancy

contains an integer overflow vulnerability. The integer overflow can be triggered by calling the function with large integer values. Finally, the fifth and last chosen challenge contains a side-channel leakage vulnerability. The information leakage occurs due to the fact that the function performs string string comparison and the running time depends on its inputs. The last chosen challenge is more typical in embedded systems.

■ **Listing 1** Vulnerable Code Snippet Containing CWE-208.

```
int is_equal(const char* a, const char* b, size_t len) {
    for (size_t i = 0; i < len ; i++) {
        if (a[i] != b[i])
            return 1;
    }
    return 0;
}
```

Listing 1 shows the source code corresponding to the fifth challenge. The problem with the code is that, the for loop will break depending on the contents of the input *a* and input *b*. Whenever the first difference is found, the for loop breaks and the function results 1. If both vectors contain the same values, the for loop will take the longest time to run, dependent on the length of the vectors. In the Sifu platform, the user is also given the information that the input *a* and *b* are of the same length, and that this length is equal to *len*.

For this example, the desired answer from the player corresponds to the code shown in listing 2. In this listing, the function does not return immediately when the first unequal values are observed. The run time of the function will be constant, and only dependent on the length of the vectors. Since the returned value of the comparison does not depend on the contents of the input vectors *a* and *b* but only on their length, no information is leaked by running the algorithms, i.e. an attacker able to manipulate one of the inputs cannot gain information about the other input by means of the time the algorithm takes to run.

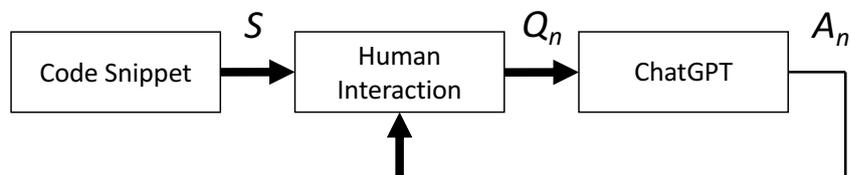
■ **Listing 2** Desired Challenge Solution.

```
int is_equal(const char* a, const char* b, size_t len) {
    if (a == NULL || b == NULL) return -1;

    int result = 0;
    for (size_t i = 0; i < len; i++) {
        result |= a[i] ^ b[i];
    }
    return result;
}
```

## 2:6 I'm Sorry Dave, I'm Afraid I Can't Fix Your Code

Figure 1 shows how the interaction was carried out with ChatGPT. For each challenge, a different code snippet  $S$  containing vulnerable code is provided. To test the algorithm, the author conducted several human interactions with the ML algorithm, for each code snippet  $S$ , corresponding to the five chosen challenges.



■ **Figure 1** Interaction with ChatGPT.

The interactions with ChatGPT consisted of a separate session of questions  $Q_n$  posed to the algorithm and its corresponding answers  $A_n$ .

■ **Table 2** Human Interactions with ChatGPT.

Nr	Question	Expected Answer
1	What is the vulnerability present in the following code snippet $S$	Correct vulnerability identification
2	What is the corresponding CWE number?	CWE number according to table 1
3	Please fix the code	Correct fix of the code
4..15	There is still a vulnerability in the code, please fix it	Improved code
>15	The code contains vulnerability XXX, please fix it	Improved code

The strategy to ask questions was the following. In the first question, we ask ChatGPT to identify the vulnerability by name. Since ChatGPT is verbose, we expected it to output a description of the problem. In the second question, we wanted to get the CWE number corresponding to the vulnerability to test if it matches our design. The design of both of these questions has the goal to identify how ChatGPT can support a software developer in finding and understanding secure code problems.

In the next phase (i.e. starting with question 3), we asked ChatGPT to fix the code, based on its previous answers. Our expectation is that the fixed code will correctly address the challenge vulnerability. We also carried out additional questions (4..14), where we claimed to ChatGPT that there were additional vulnerabilities and that ChatGPT should fix them. The goal of this last question was to determine how far could the algorithm could detect further problems and iteratively improve the code. Finally, on the sixteenth question, we claimed to ChatGPT that the code contained the intended vulnerability, and that ChatGPT should fix it.

The experiments were carried out through the online interface of ChatGPT on the 15th January 2023. It consisted of a total of 43 interactions with the ChatGPT user interface, corresponding to 5 for CWE-121, 5 for CWE-758, 5 for CWE 242, 11 for CWE-190, and 17 for CWE-208. The version of ChatGPT reported in the user interface was “ChatGPT 9 Jan Version”.

## 4 Results

Table 3 shows a summary of the challenges and their corresponding identified vulnerability by ChatGPT. Since the CWE identified by ChatGPT was not matching exactly the CWE from the challenge, we decided to compare the solution based on the proximity of the answer

from ChatGPT. We concluded that, for the first three challenges, the vulnerabilities that were identified were corresponding to a specialization of the problem. While the challenge CWE considered the general case, ChatGPT was more precise in its findings. Therefore, we concluded that the answer from ChatGPT was acceptable in those circumstances.

■ **Table 3** Vulnerabilities Identified by ChatGPT.

ID	Designed Challenge Vulnerability	ChatGPT Identified Vulnerability	Assessment is Acceptable?	Description
1	CWE-121	CWE-120	YES	Correctly identified that buffer overflow occurs due to copying string
2	CWE-758	CWE-835	YES	Correctly identified an infinite loop due to undefined behavior
3	CWE-242	CWE-121	YES	Technically correct assessment, but failed to identify that gets is deprecated
4	CWE-190	CWE-835	NO	Assessment of problem misses the major issue, in favor of smaller detail
5	CWE-208	CWE-121	NO	While technically correct, the assessment is wrong due to missing context

For the fourth and fifth challenge (CWE-190 and CWE-121), ChatGPT did not correctly identify the problem with the code. For the CWE-190 challenge, ChatGPT found another problem, which although correct, was not the major problem with the code. Since the major problem with the code was not identified, we decided not to accept ChatGPT's solution. For the CWE-208 challenge, ChatGPT completely missed the problem. Even though we asked several times (4.15) to find and fix vulnerabilities in the code, ChatGPT could not identify it. In the 16th iteration, we told ChatGPT that the code had the CWE-208 vulnerability and that ChatGPT should solve fix the code.

Due to the interesting nature of this special case, we present the solution from ChatGPT at question 16. Listing 3 shows the solution presented by ChatGPT. In this case, ChatGPT used the CRYPTO\_memcmp from OpenSSL to solve the problem

■ **Listing 3** ChatGPT Solution to CWE-208 Challenge (using OpenSSL).

```
#include <openssl/crypto.h>

int is_equal(const char* a, const char* b, size_t len) {
    if(a == NULL || b == NULL) return -1;
    size_t a_len = strlen(a, len);
    size_t b_len = strlen(b, len);
    if (len == 0 || len > a_len || len > b_len) {
        len = a_len < b_len ? a_len : b_len;
    }
    if (a[a_len] != '\0' || b[b_len] != '\0') {
        return -1;
    }
    return CRYPTO_memcmp(a, b, len);
}
```

Since we were not expecting that ChatGPT would use an external library, on iteration 17 we asked ChatGPT to provide us with a solution without using OpenSSL. The answer from ChatGPT is presented in Listing 4.

■ **Listing 4** ChatGPT Solution to CWE-208 Challenge (without using OpenSSL).

```
int is_equal(const char* a, const char* b, size_t len) {
    if(a == NULL || b == NULL) return -1;
    size_t a_len = strlen(a, len);
    size_t b_len = strlen(b, len);
    if (len == 0 || len > a_len || len > b_len) {
        len = a_len < b_len ? a_len : b_len;
    }
}
```

```

    if (a[a_len] != '\0' || b[b_len] != '\0') {
        return -1;
    }
    int result = 0;
    for(size_t i = 0; i < len; i++) {
        result |= a[i] ^ b[i];
    }
    return result;
}

```

In this listing, we can see that the desired solution (see listing 2) is present in the code. However, additional tests are performed in relation to the length of the input arrays, which are unnecessary, given the context in which the function is used. Furthermore, ChatGPT has assumed that both inputs represent a string and therefore tests for the case that the inputs are null-terminated. We note that these modifications to the source code introduced by ChatGPT change the semantic of the function.

## 5 Discussion

According to our experience in teaching secure coding in the industry, and as a result of the interactions taken with ChatGPT, we can conclude that this ML algorithm has a very high potential to be used to assist software developers in writing secure code. In more than 60% of the code that we provided to ChatGPT, it was able to correctly identify the problem with the source code, and to provide a good fix., This was very surprising since, as briefly discussed in the introduction, the problem of secure coding leads to non-decidable problems. We hypothesize that the reason for this success has to do with the fact that the code snippets that were supplied to ChatGPT were relatively small (i.e. less than 40 lines of code). Further investigations would be needed on the efficiency of ChatGPT in identifying software vulnerabilities in large code basis.

Another surprise was the fact that the explanations about the problems contained in the challenges was matching very well with the actual problem. ChatGPT's explanation was not only 3/5 of the time correct, but it was also precise in the identification and explanation of the problem. We see this as a clear advantage for ChatGPT as a teaching tool. Nevertheless, due to the fact that only a small number of snippets were tried, and that the code snippets were small in size, more investigation needs to be carried out to fully understand the usage of ChatGPT for teaching purposes.

Another point that surprised us was the fact that, not only could ChatGPT interpret the code, but could also suggest fixes to it. In particular, some of the code fixes suggested by ChatGPT could be considered to be creative. This is a clear indicator of how advanced the implementation of the algorithm is.

However, several limiting factors have also been found while interacting with ChatGPT.

We were surprised of how good the model is, but we also found limitations to its use, as it is lacking on some aspects. In the following we summarize the major aspects that we found that can limit the usage of this technology:

**Missing Context.** ChatGPT lacks the context in which the code is being used. This can lead to superfluous corrections and bug fixes which are not necessary due to the boundary conditions

**Change Semantics.** One major problem that was identified in the fourth challenge was that the solution given by ChatGPT changed the semantic of the code in a very subtle way; this means that code before fixing and after fixing can behave slightly different. This

can be a strong deterrent factor to use this technology in practice (e.g. for safety-critical systems). Changes in the code should be semantic-preserving and ChatGPT currently does not guarantee this

**Code Complexity.** Code produced by ChatGPT has the potential to have more computational complexity than the original (vulnerable) code. This can potentially introduce computational inefficiencies, which is a critical aspect for real-time systems

**Code Maintainability.** Code produced by ChatGPT lacks maintainability characteristics, e.g. due to increased complexity or missing comments

**Limited Learning.** ChatGPT has been tuned with data up to 2021. This means that potential new threats and vulnerabilities that have been found since then might not be well processed by the system. Further investigations would need to be carried out to test the algorithm in this circumstances

**Learning Interference.** ChatGPT can learn from user interactions. For the algorithm to be used in a professional environment (especially in safety-critical systems and critical infrastructures), some protections need to be added such that the algorithm behind ChatGPT cannot learn incorrect data and therefore does not give bad answers.

Combining our experience, previous research , and our experiment with ChatGPT, we conclude with a reinforcement of the conclusions done in [8], and in [11]. In particular, we reinforce the conclusion that using an AI/ML engine can be an excellent approach for teaching and raising awareness of secure coding in software. Further research could integrate ChatGPT into the Sifu platform to further validate the approach with real-world scenarios and software developers in the field.

This work shows the potential that ChatGPT has to be used not only as a teaching tool, but also as a tool to assist professional software developers in the industry. We think that the tool can assist software developers to think outside the box and find creative new solutions to complex problems. One example of the usage of generative AI technology to assist software developers write code is GitHub's Copilot [12]. However, as per conclusions on the present work, while Copilot can help software developers write code faster, further investigation is needed to understand the extent to which this software development model can introduce or eliminate the introduction of vulnerabilities in software.

Additionally, careful reflection and care must be carried out when using AI models in an industrial environment, since the model can potentially learn also from input which is provided to it. This could potentially lead to serious leakages of information to the wide public, e.g. on software weaknesses in the products and services offered by the company. Therefore, according to our experience, we consider in-house usage of AI for assisting software development to be the appropriate means to use the technology in an industrial context.

Finally, we would like to reflect on the possible usage of ChatGPT as a means to understand the output of SAST tools. Our experience has shown that software developers do not always understand the output that is provided by SAST tools and, therefore, cannot recognize the corresponding vulnerability in software. We think that ChatGPT could be used to analyse the result of these tools and to aid software developers to understand this output and therefore to write better code. However, further research is needed to validate this point.

In conclusion, we would like to highlight a further possible problem which is related to software plagiarism. ChatGPT learns from many different sources (containing many different types of software licenses) and summarizes the output of these while processing the answers to the questions. One potential problem and danger of using ChatGPT is the fact that, since it can produce code that is equal to some random source, or may be considered derivative work, this could lead to potential legal problems.

Our experience and previous research has shown that AI-based technologies can be used to effectively raise awareness of software developers, both in an industrial setting and also in the academia. Nevertheless, while we observe good indicators for ChatGPT as tool to teach cybersecurity, we are not entirely convinced that this is the case, especially for the industry, as not only the answers provided by the model can be partially wrong, it is not clear the status of the copyright infringement of the provided solutions.

To wrap-up our final conclusions, we add a text generated by ChatGPT itself:

*ChatGPT is not able to fix code or help developers write secure code. While it is able to understand and respond to natural language inputs related to coding and cybersecurity, it does not have the capability to understand or execute code. It can provide general information and suggestions based on the information it has been trained on, but it cannot guarantee that the suggestions will be accurate or complete. Additionally, ChatGPT is not able to identify or fix specific vulnerabilities in code.*

*While ChatGPT can be a useful tool for developers, it should not be relied upon as a sole solution for ensuring the security of code. Developers should use a combination of techniques, such as threat modeling, penetration testing, and code review, to identify and fix vulnerabilities in their code. Additionally, it is important for developers to stay informed about the latest threats and vulnerabilities, and to follow secure coding practices.*

## **5.1 Threats to Validity**

The present work is based on the ChatGPT version from January 13 2023, with the training model GPT-3. As the model is being rapidly improved, part of our results might not be reflected in later versions of the improved model. This can potentially limit or invalidate part of the conclusions in the present work. While the authors' believe that the generative AI technology will experience a significant improvement, leading to potential better results as those hereby presented, we also claim that there are fundamental theoretical limitations that will certainly impose limits to its usefulness and practical applicability.

## **6 Conclusion**

Software security is not only an important topic of software development, it has been gaining much attention over the last years. The reason for this, is the fact that there is an increasing number of cybersecurity incidents taking place that have negative consequences for society in general. A possible root-cause of cybersecurity incidents is related with poor coding practices. Therefore, to reduce the number of incidents, software developers should know and employ best practices while developing software. These practices are generally taught either in the university, or during an internal training in the industry.

In this paper, we look at the potential of using Machine Learning algorithms to both assist software developers to write secure code, but also as a tool to raise awareness of secure coding. Previous work has shown that artificial intelligence techniques can be successfully used to train software developers in secure coding guidelines. This paper presents a preliminary exploration of the usage of ChatGPT to raise secure coding awareness. Our work follows not only from the experience of the authors, but also on their extensive work in the field. In this paper, we reflect on the advantages and disadvantages of the usability of ChatGPT or similar algorithms and show that, while ChatGPT has a clear potential to be used as an aid to software development, there are some limitations to its usage. In further work, the authors would like to integrate ChatGPT with CyberSecurity Challenges – a serious game to raise awareness of secure coding guidelines of software developers in the industry. Our preliminary research presented in the present work lead us to expect good results of this integration.

---

**References**

---

- 1 Yasemin Acar, Christian Stransky, Dominik Wermke, Charles Weir, Michelle Mazurek, and Sascha Fahl. Developers Need Support, Too: A Survey of Security Advice for Software Developers. *2017 IEEE Cybersecurity Development (SecDev)*, pages 22–26, September 2017. IEEE Computer Science, Engineering. doi:10.1109/SecDev.2017.17.
- 2 Bushra Aloraini, Meiyappan Nagappan, Daniel German, Shinpei Hayashi, and Yoshiki Higo. An Empirical Study of Security Warnings From Static Application Security Testing Tools. *Journal of Systems and Software*, 110427(158):1–25, December 2019. Elsevier, Amsterdam, Nederland. doi:10.1016/j.jss.2019.110427.
- 3 Roberto Bagnara, Abramo Bagnara, and Patricia M Hill. Coding guidelines and undecidability. *arXiv preprint*, 2022. arXiv:2212.13933.
- 4 Bundesamt für Sicherheit in der Informationstechnik. BSI IT-Grundschutz-Katalog. Technical report, Bundesamt für Sicherheit in der Informationstechnik, Reguvis Fachmedien GmbH, Köln, Germany, April 2016. 15. ed, BSI. URL: [https://download.gsb.bund.de/BSI/ITGSK/IT-Grundschutz-Kataloge\\_2016\\_EL15\\_DE.pdf](https://download.gsb.bund.de/BSI/ITGSK/IT-Grundschutz-Kataloge_2016_EL15_DE.pdf).
- 5 Carnegie Mellon University. Secure Coding Standards. Software Engineering Institute, Online, Accessed 19 March 2019. URL: <https://wiki.sei.cmu.edu/confluence/display/seccode>.
- 6 MITRE Corporation. Common Weakness Enumeration. Online, Accessed 4 July 2019. URL: <https://cwe.mitre.org/>.
- 7 Ralf Dörner, Stefan Göbel, Wolfgang Effelsberg, and Josef Wiemeyer. *Serious Games: Foundations, Concepts and Practice*. Springer International Publishing, 1 edition, September 2016.
- 8 Tiago Gasiba, Ulrike Lechner, and Maria Pinto-Albuquerque. Sifu - A CyberSecurity Awareness Platform with Challenge Assessment and Intelligent Coach. *Special Issue of Cyber-Physical System Security of the Cybersecurity Journal*, pages 1–23, October 2020. SpringerOpen, Online. doi:10.1186/s42400-020-00064-4.
- 9 Tiago Gasiba, Ulrike Lechner, Maria Pinto-Albuquerque, and Daniel Mendez Fernandez. Awareness of Secure Coding Guidelines in the Industry - A First Data Analysis. In Guojun Wang, Ryan Ko, Md Zakirul Alam Bhuiyan, and Yi Pan, editors, *TrustCom 2020: International Conference on Trust, Security and Privacy in Computing and Communications*, pages 345–352, December 2020. IEEE, Guangzhou, China. doi:10.1109/TrustCom50675.2020.00055.
- 10 Tiago Gasiba, Ulrike Lechner, Maria Pinto-Albuquerque, and Daniel Mendez. Is Secure Coding Education in the Industry Needed? An Investigation Through a Large Scale Survey. In Hakan Erdogmus and Ana María Moreno, editors, *43rd International Conference on Software Engineering*, pages 1–12, May 2021. . URL: <https://arxiv.org/abs/2102.05343>.
- 11 Tiago Espinha Gasiba. *Raising Awareness on Secure Coding in the Industry through CyberSecurity Challenges*. PhD thesis, Universität der Bundeswehr München, 2021. URN: urn:nbn:de:bvb:706-7860. URL: [https://athene-forschung.unibw.de/85049?query=gasiba&show\\_id=140142](https://athene-forschung.unibw.de/85049?query=gasiba&show_id=140142).
- 12 GitHub. Copilot. Online, Accessed 15 June 2023. URL: <https://github.com/features/copilot>.
- 13 Katerina Goseva-Popstojanova and Andrei Perhinschi. On the Capability of Static Code Analysis to Detect Security Vulnerabilities. *Information and Software Technology*, 68:18–33, December 2015. Butterworth-Heinemann, Newton, MA, USA. doi:10.1016/j.infsof.2015.08.002.
- 14 Jacob A. Harer, Louis Y. Kim, Rebecca L. Russell, Onur Ozdemir, Leonard R. Kosta, Akshay Rangamani, Lei H. Hamilton, Gabriel I. Centeno, Jonathan R. Key, Paul M. Ellingwood, Marc W. McConley, Jeffrey M. Opper, Sang Peter Chin, and Tomo Lazovich. Automated Software Vulnerability Detection with Machine Learning. *CoRR*, abs/1803.04497, 2018. arXiv:1803.04497.

## 2:12 I'm Sorry Dave, I'm Afraid I Can't Fix Your Code

- 15 International Electrotechnical Commission. IEC 62443-4-1 – Security for industrial automation and control systems - Part 4-1: Secure product development lifecycle requirements. Technical report, International Electrotechnical Commission, Geneva Switzerland, January 2018. .
- 16 International Organization for Standardization. ISO/IEC 25000:2014 – Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE. Technical report, International Organization for Standardization, Geneva, CH, March 2014. Software and Systems Engineering. URL: <http://iso25000.com/index.php/en/iso-25000-standards>.
- 17 Akram Louati and Tiago Gasiba. Source Code Vulnerability Detection using Deep Learning Algorithms for Industrial Applications. In *The Second International Conference on Ubiquitous Security (UbiSec 2022)*, pages 1–19, December 2022. .
- 18 Open Web Application Security Project. OWASP Top 10. Online, Accessed 15 July 2017. URL: <https://tinyurl.com/yyb8wcv9>.
- 19 OpenAI LP. ChatGPT. Online, Accessed 23 January 2023. URL: <https://chat.openai.com/>.
- 20 Tosin Daniel Oyetoyan, Bisera Milosheska, Mari Grini, and Daniela Soares Cruzes. Myths and Facts About Static Application Security Testing Tools: An Action Research at Telenor Digital. *International Conference on Agile Software Development*, pages 86–103, May 2018. Springer, Cham. doi:10.1007/978-3-319-91602-6\_6.
- 21 Suri Patel. 2019 Global Developer Report: DevSecOps finds security roadblocks divide teams. Online, Accessed 18 July 2020. URL: <https://about.gitlab.com/blog/2019/07/15/global-developer-report/>.
- 22 Tim Rietz and Alexander Maedche. LadderBot: A Requirements Self-Elicitation System. *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pages 357–362, September 2019. IEEE, Jeju, South Korea. doi:10.1109/RE.2019.00045.
- 23 Gaigai Tang, Lianxiao Meng, Shuangyin Ren, Weipeng Cao, Qiang Wang, and Lin Yang. A Comparative Study of Neural Network Techniques for Automatic Software Vulnerability Detection. *CoRR*, abs/2104.14978, 2021. arXiv:2104.14978.

# LCSMAR, an AR Based Tool to Inspect Imperative Programs

Luis Carlos Martins ✉🏠

ALGORITMI Research Centre / LASI, DI-University of Minho, Braga, Portugal

Lázaro Vinicius Lima ✉🆔

Department of Informatics, Instituto Federal de Brasília, Brasil

Pedro Rangel Henriques ✉🏠🆔

ALGORITMI Research Centre / LASI, DI-University of Minho, Braga, Portugal

---

## Abstract

LCSMAR is a Learning Resource that takes advantage of Augmented Reality in order to promote the development of Computational Thinking among students who are starting to dive in to the world of computer programming. Students can write code in a imperative programming language and, with the help of their mobile phone, they can visualize and analyze the execution of the code they developed, seeing how variable and data structures change over time with each instruction. Augmented Reality tools allow the visualization of abstract concepts that are often misunderstood and that cause misconception among students, which in term should help students develop the abilities to understand and use these abstract concepts, such as data structures, in other areas of application.

**2012 ACM Subject Classification** Computing methodologies → Mixed / augmented reality

**Keywords and phrases** Augmented Reality, Learning Resources, Computer Programming, Computational Thinking

**Digital Object Identifier** 10.4230/OASICS.ICPEC.2023.3

**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

Augmented Reality (AR) is a technology that has often been associated with the world of entertainment and gaming[8]. However, in recent years, its has gained more attention for its educational purposes and has started to being used as a Learning Resource (LR)[2]. The use of an AR in an educational environment gives students new tools and possibilities that a normal, tradition classroom cannot give, thus making their new learning experiences less stale and mundane, creating opportunities for new AR tools to be created.

Computational Thinking (CT) is the thought process that going into formulating problems and the solution for them and its an essential skill to have in the 21st Century[9]. CT enclose a range of concepts that define what it is, such as algorithmic thinking, decomposition, patter recognition and abstraction. One of the core concepts of CT is abstraction, that involves focusing on the essential features of a problem or system while ignoring irrelevant details, but it can be a challenging concepts to grasp, specially for students learning any new concepts that doesn't have a concrete representation and that may cause confusions and misconceptions between students.

Since AR displays computer generated information onto the real world, it is possible to the same with abstract concepts, displaying them in a virtual environment that students could visualize and interact with them in a way that isn't possible in tradition ways, thus being an important technology to be used when teaching in the modern world.



© Luis Carlos Martins, Lázaro Vinicius Lima, and Pedro Rangel Henriques; licensed under Creative Commons License CC-BY 4.0

4th International Computer Programming Education Conference (ICPEC 2023).

Editors: Ricardo Alexandre Peixoto de Queirós and Mário Paulo Teixeira Pinto; Article No. 3; pp. 3:1–3:9

OpenAccess Series in Informatics



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

Language-Specific Code Simulation with a Mobile using AR (LCSMAR) is a mobile application with AR where students can develop their own pieces of code in an imperative language that then can be visualized in an AR environment step-by-step, thus visualizing the execution of the code, the changes to the variables and data structures in real-time. This approach aims to help students understand the underlying functionalities of basic data structures, such as lists, queues and stacks and how to manipulate them in a correct way, while at the same time visualizing and learning how they work.

In this article, we present the development of an AR-based Learning Resource called LCSMAR (Section 5) after discussing in detail its architecture and desired functionalities (Section 4). Before that the main areas of research involved in this project, Computational Thinking and Augmented Reality, are discussed in Sections 2 and 3 respectively. Section 6 closes the paper.

## 2 Computational Thinking

As defined by Wing [9], Computational Thinking is the thought process that goes into formulating problems, being a collection of skills based on the principles of Computer Science, therefore being one of the most essential skills of the 21<sup>st</sup> century.

It is a very important skill that, such as reading or writing, must be taught to the younger population at an early age, in order to allow them to develop the necessary skills, not only for CS but also for other areas of science, being mainly focused on the aspect of systematizing, representing, analyzing and problem solving skills.

Just like any other skill, it is important to develop and train it and to do so it is needed specialized tools and LR to help teachers teach their students to develop CT.

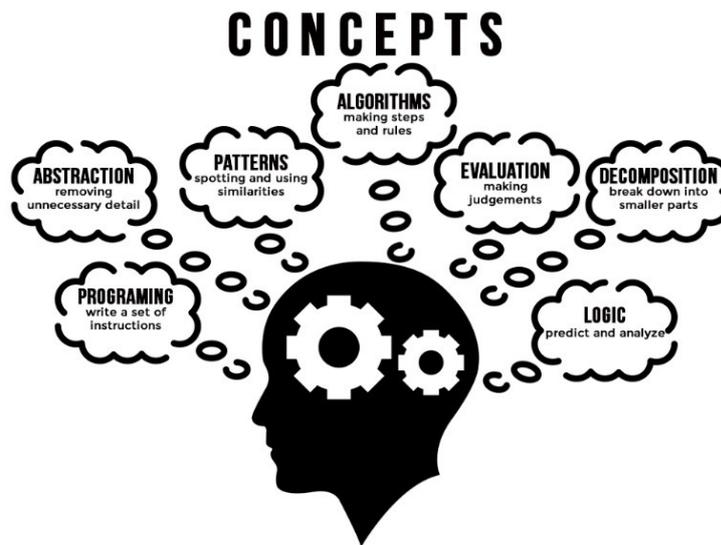
One of the ways to effectively train CT is through the use of Learning Resources (LR) that, as the name suggests, are tools designed to assist teachers teaching and students learning. These resources, which can either be physical or digital devices, allow students to not only gain knowledge during classes but also allow them to develop said knowledge, therefore making the learning experience interesting, appealing and modern, thus stimulating their interests in acquiring new skills [3].

In order to train CT it is necessary to use LR to develop this thought process. As CT requires a wide range of different skill sets, including abstraction and problem decomposition (Figure 1), it is crucial to have appropriate resources to develop each one of these skills, facilitating students' development of CT skills. These tools must be easy to understand and to use but also must have high potential for students to develop their skills.

There are two main kinds of LRs: *plugged* or virtual resources that must be used with electronic equipment and devices such as smartphones and cameras and *unplugged* resources that don't. Each one of these types offer different advantages that depend on the environment that they are being used.

The initiative CS Unplugged <https://www.csunplugged.org/en/> is a collection of many different *unplugged* activities like card games, puzzles and challenges that aim to build the ground work necessary for students to dive into Computer Science (CS). This initiative not only provides materials for student to learn concepts about CS but also provides teachers with content to teach during classes, thus providing students with a favourable environment to develop and train their CT skills.

One *plugged* LR that's very widely known and used in the work is *Scratch* ([7]) that is used to introduce student to programming and also to CT. There is a big community around *Scratch*, where students can share their creations with other students around the world, as well as tinker and modify other's projects to learn how they work. This tool allows students



■ **Figure 1** Computational Thinking Concepts that can be trained.

to use blocks of code that can be used interchangeably to make their own creation, while developing and using computer programming concepts such as conditionals, loops and events, while at the same time that they develop CT related skills.

Analyzing the success and impact that *Scratch* has in students, where they get motivated and interested in developing their own applications and games, its possible to conclude that *unplugged* activities have the power to make students more engaged in learning, showing a need for new pieces of technology that provide students with material to learn new sets of skills, starting with CT.

Thus there is a need for new and innovative LR that allow students to learn new concepts and skills that are easy and intuitive to use, while at the same time being effective at delivering its promise of teaching and engaging students.

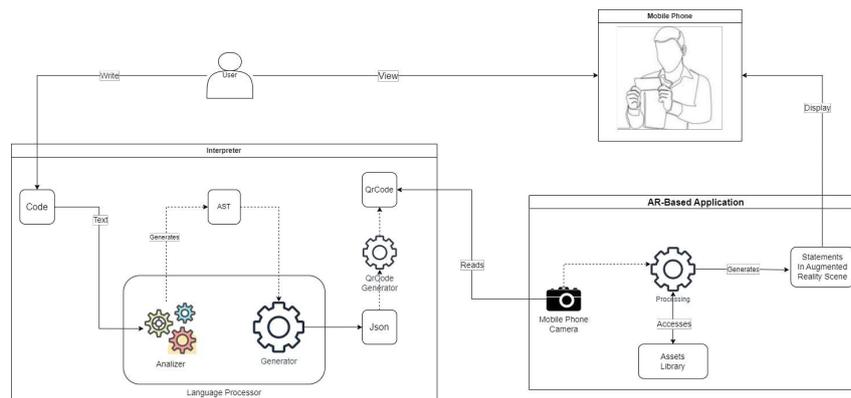
### 3 Augmented Reality

Virtual Environment (VE) technologies are becoming more prominent in our world, becoming a key technology for the near future and already having a lot of uses in our society, helping and enhancing our daily tasks.

One of the most popular and used VE technologies is Augmented Reality, being one of the spectrums of the Mixed Reality continuum, as presented by [6]. When using VR the user is completely immersed in a fully virtual environment, separate from the real world, through the use of head-mounted displays and sensory devices, while when using AR devices, the user experiences the real world enhanced by information generated by these devices, adding digital information to the real world. While VR provides a complete virtual experience, AR allows for a mix between real and virtual world.

AR is the enrichment of a real environment by bringing virtual information into the world using technological devices, providing users with computer generated texts, images and/or virtual objects that are superimposed in the real objects that they see, bridging these distinct worlds [1]. AR aims to explore the use of vision and hearing in these environments but it is not exclusive to these senses, as presented by Azuma [1], where all the others senses, such as tact and smell can also be used and explored.

### 3:4 LCSMAR, an AR Based Tool to Inspect Imperative Programs



■ Figure 2 System Architecture.

To take advantage of AR, there are multiple types of devices available to be used that can be subdivided in the following categories. Firstly, head-mounted displays (HMD) are designed to be worn on the head and a projection is displayed in the user's field of view. This type of AR is mainly used in a industrial setting where worked need hand-free access to information. Secondly, handheld devices that are much more portable than the one above since it can be used with the help of a smartphone to display, using the phones cameras, AR information superimposed into the real world. Lastly, using spatial displays, the computer generated information can be displayed directly into the real world without the need for any display. Some examples of these kinds of displays are video-projector or holograms, allowing the virtual world to extend to a wider amount of people simultaneously.

Even tho that at first sight AR look more like a piece of entertainment over anything else, it is a very useful tool to help teachers and students at the same time. With AR its possible to captivating the students attention and motivation while at the same time promoting their own development of skills. The work of [4], CodeCubes, is a game with an AR interface developed to teach scientific principles of computational thinking. In order to solve the tasks, students need to explore, experiment, and interact with CodeCubes putting to practice the trial-and-error method. This approach aims at introducing basic programming concepts to children through experimentation. It combines physical paper cubes with AR technology for teaching basic programming concepts.

Another tool that uses AR in the objective to teach students is the work of [5], stating that programming is an effective way to promote the development of CT, presenting a tangible programming tool that utilizing AR technology for kids, helping them create their own programs by positioning programming blocks and executing code with a mobile device. Through this, kids can learn fundamental programming concepts, such as parameters, loop logic, debug and so on. Following the authors' tests in children, they found that they were engaged and enjoying the game, indicating that these tools are conducive for them to understand several computer concepts. The authors also realized that children are more likely to ignore feedbacks that are received through plain text.

## 4 Architecture

The developed Learning Resource consist of two main components (Figure 2, each serving a unique purpose and important role in the environment of the system.

The first component Language-Specific Code Interpreter that's written in Python and utilizes the Lark library to do so. This component is responsible for analyzing and interpreting code in a custom imperative language. This library was chosen because it is very flexible and

powerful parsing library for Python, allowing to create the custom imperative language with ease, making it an ideal choice for the project. This component take the code that a user writes, analyzes and executes it and create information that will be transmitted through a Qr-Code and then processed by the second component in the mobile app.

The second component is an AR mobile app that utilizes the Vuforia AR software. In order to gather the processed information from the first component, the ZXing library is used in order to read the generated Qr-Code, which will then be processed in order to layout the initial virtual environment, where the users can see the execution of their written code step by step. The choice to use Vuforia as the AR platform for the mobile app was driven by the fact its a robust piece of software that allows for easy development of AR based programs.

In order to create the Virtual Environment a the AR mobile app uses the data created by the Interpreter in order to display the statement in the AR scene. This data represents a JSON with all the necessary information to be able to advance the code step-by-step while at the same time allow for the user to step back into the execution of the code, only needing to follow the flow of information represented in the JSON itself, thus being a solution not very computationally intensive.

One of the key goals of this project it to develop and AR app that is able to create a tool that can be used in schools that is both easy to use and highly intuitive while also being engaging for the user allowing them to easily visualize the execution of the code they've written step-by-step. The interface was designed to be as easy to use as possible with a clear and simple interface to navigate. Creating an engaging experience was one of the main priorities during the development of the app, with the objective of creating visual that are easy to understand while at the same time being engaging. Despite being able to use HMD displays to take advantage of AR, we decided to stick with the use of smartphones for the fact that they are easier to use, more accessible, an already build-in camera and many students already have one, so it can be used by a larger number of the population.

In order to transmit information from one component to another, many solutions where considered to do so, but the one that made more sense was through the use of QR Codes. QR Codes are two-dimensional barcodes that can easily be read by smartphones. They can stored any type of information from links to website, product details, etc, but can also save binary data that then can be used. So with their help, its possible to transmit information from one component to another with the need to setup additional infrastructure. Since QR Codes have a limited amount of data that can be stored, in order to fit the most amount of data in it as possible, the instructions generated by the interpreter component are compressed before being transformed into a QR Code, thus allowing to storage large amounts of data into it.

## 5 LCSMAR

The developed LR, as said before, is divided into two components, the first one, the interpreter, it takes the code that the students write in a imperative programming language, interprets it and generates information that can then be used by the second component to lay-out the execution of the code written in a AR environment.

In order to make these two components more independent from each other by not creating a direct way of communication between them, there was a need to find another possible way to transmit information from one component to another, and to do so we used QR Codes, since they are easy to generate and scan, and can be read by almost every modern smartphone. Thus, by using QR Codes there is no need to create server making it an LR that can be used without the need for an internet connection.

### 3:6 LCSMAR, an AR Based Tool to Inspect Imperative Programs

■ **Listing 1** Example code of the developed imperative language.

```
1  VARS
2  i, evenSum, oddSum: int;
3  listOdd, listEven: List(int);
4  STATS
5  evenSum = 0;
6  oddSum = 0;
7  for (i = 1; i <= 10; i++){
8      if (i % 2 == 0){
9          listEven.insert(i);
10     } else {
11         listOdd.insert(i);
12     }
13 }
14 while(listEven.len > 0){
15     evenSum += listEven.remove();
16     oddSum += listOdd.remove();
17 }
18 END
```

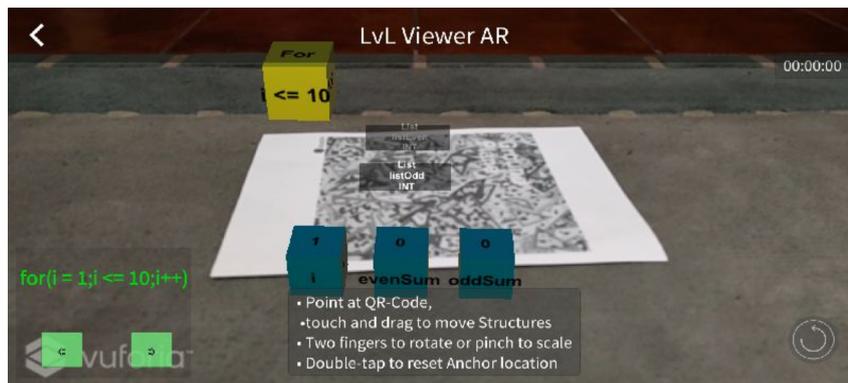
The main focus of this project was developing an AR application that its easy to use, engaging, appealing and that allowed students to understand better how their code works and how to manipulate and use data structures. Some of the features of the developed application are, visualizing the written code, advancing or stepping back each line of code, see the changes that occur in each variable and data structure, know if the current line of code is inside a conditional block, knowing if the condition is either true or false and know if a line is inside a cycle.

The first step to use this LR is to write the code that will generate a QR Code, that then can be read using the mobile phone. To do so, the users will write their code in a web-browser based application that is able to generate the QR Code thanks to the information generated by the Interpreter (Figure 3).



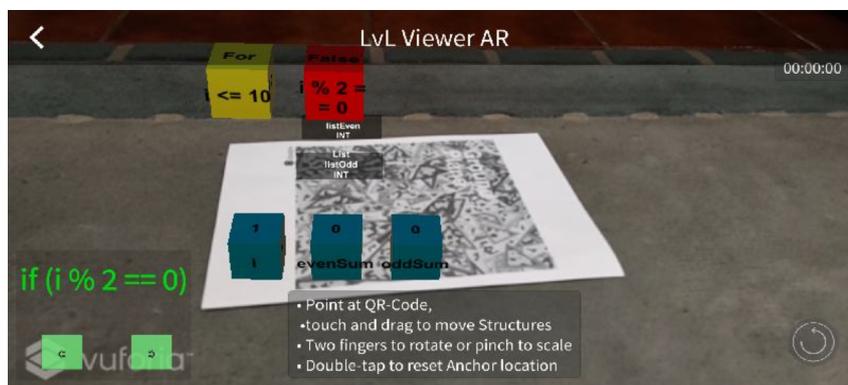
■ **Figure 3** Web App where students can write their code.

After reading the generated QR Code, the starting scene is generated, with the starting variables and the empty data structures, which can then be advanced step by step using the UI. The piece of code 1 will be used to show the features of the AR application. The code puts the numbers between 1 and 10 in a corresponding List of either even or odd numbers, and then it calculates the sum of each, storing these values in their corresponding variable. In the Figure 4 the variables and their states can be seen represented in the blue cubes, that have the name of the variable as well as its state.



■ **Figure 4** Code execution representing a for cycle.

When the code reaches the for cycle a new block appear representing it, showing the loop condition. In a similar way a block that represents the if condition can either be red (Figure 5 )when the condition is false or green if the condition is true.



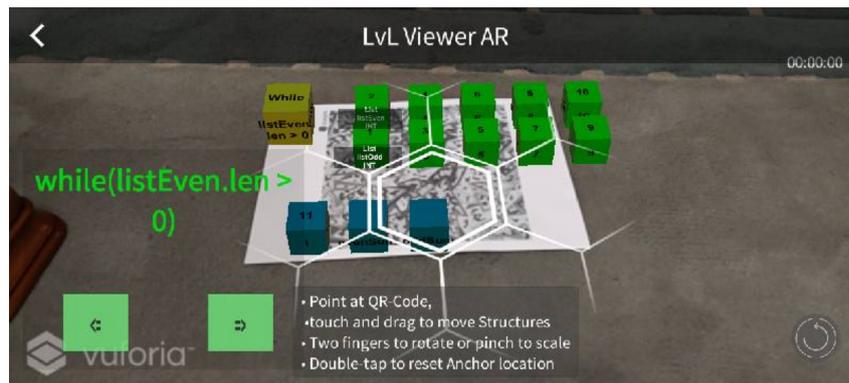
■ **Figure 5** Code execution representing an if condition.

As the code is executed the content of each data structure changes, adding more blocks to each one, representing each element as can be seen in the Figure 6.

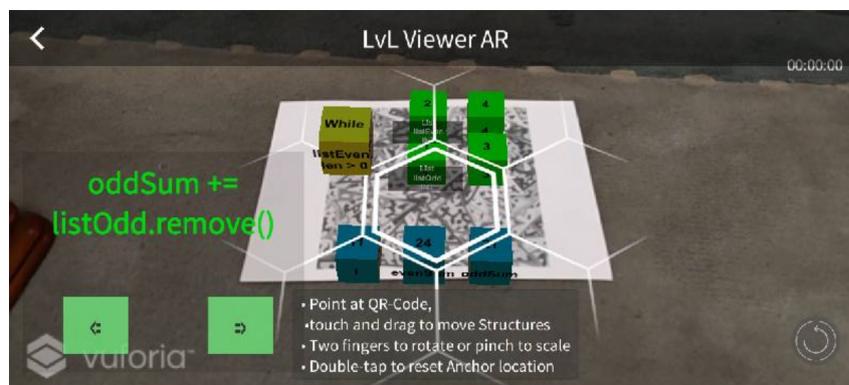
Then inside a while loop (Figure 7) that is represented by another colored block, the elements of each data structure will be removed from them and then they are added to the respective variable, thus calculating the sum of the numbers.

This LR allow for students to have a different approach into learning how code works and the intricacies that are the manipulation and use of data structures. By using a variety of teaching methods and tools, such as visual aids, interactive exercises, and real-world examples, students can gain a deeper understanding of the concepts and develop problem-solving skills that they can apply in their future careers. Additionally, a well-designed LR can help students stay engaged and motivated throughout the learning process, which can improve their overall retention and mastery of the material.

### 3:8 LCSMAR, an AR Based Tool to Inspect Imperative Programs



■ **Figure 6** Code execution representing the state of the data structures.



■ **Figure 7** Code execution representing removal of elements from data structures during a while loop.

## 6 Conclusion and Future Work

In this article, we have presented LCSMAR, an innovative LR that uses AR to enhance students' understanding of important programming concepts and data structures. We have described the two main components of LCSMAR: the interpreter, which generates information about the code written by students; and the AR application, which uses the information generated by the previous component to lay out the execution of the code in an AR environment. At the best of our knowledge, this approach is new and can pave the way to the development of original tools to help the learning process enabling the production of nice visualizations of formal specifications or programs provided at the input. We believe that we can aid Programming teachers to improve the success of their courses. The initial development phase of LCSMAR focused on creating an easy-to-use, engaging, and visually appealing AR application that allows students to visualize step-by-step the execution of his source program emphasizing the data structures manipulation.

While the tests conducted in classrooms have shown promising results, we plan to conduct further experiments with students and educators to refine and improve LCSMAR. By incorporating feedback from users, we aim to create an effective and comprehensive learning resource that can be used to teach programming and data structures in a more engaging and interactive way.



■ **Figure 8** AR application being used.

LCSMAR here discussed is a component of a bigger project that aims at the creation of AR-based Learning Resource to train Computational Thinking. Our idea is to define a strategy to develop in a systematic way, guided by the description of the skills to train, such kind of tools.

---

## References

- 1 Ronald T Azuma. A survey of augmented reality. *Presence: teleoperators & virtual environments*, 6(4):355–385, 1997.
- 2 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.
- 3 Rona Bušljeta. Effective use of teaching and learning resources. *Czech-Polish historical and pedagogical journal*, 5(2), 2013.
- 4 Bárbara Cleto, Cristina Sylla, Luís Ferreira, and João Martinho Moura. Codecubes: Coding with augmented reality. In *First international computer programming education conference (ICPEC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
- 5 Qiao Jin, Danli Wang, Xiaozhou Deng, Nan Zheng, and Steve Chiu. Ar-maze: a tangible programming tool for children based on ar technology. In *Proceedings of the 17th ACM Conference on Interaction Design and Children*, pages 611–616, 2018.
- 6 Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality: A class of displays on the reality-virtuality continuum. In *Telemanipulator and telepresence technologies*, volume 2351, pages 282–292. Spie, 1995.
- 7 Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.
- 8 Karen Schrier. Using augmented reality games to teach 21st century skills. In *ACM SIGGRAPH 2006 Educators Program, SIGGRAPH '06*, pages 15–es. Association for Computing Machinery, New York, NY, USA, 2006. doi:10.1145/1179295.1179311.
- 9 Jeannette M Wing. Computational thinking benefits society. *40th anniversary blog of social issues in computing*, 2014:26, 2014.



# Can a Content Management System Provide a Good User Experience to Teachers?

Yannik Bauer  

DCC – FCUP, Porto, Portugal  
CRACS – INESC TEC, Porto, Portugal

José Paulo Leal   

CRACS – INESC TEC, Porto, Portugal  
DCC – FCUP, Porto, Portugal

Ricardo Queirós   

CRACS – INESC TEC, Porto, Portugal  
uniMAD – ESMAD, Polytechnic of Porto, Portugal

---

## Abstract

The paper discusses an ongoing project that aims to enhance the UX of teachers while using e-learning systems. Specifically, the project focuses on developing the teacher’s user interface (UI) for Agni, a web-based code playground for learning JavaScript. The goal is to design an intuitive UI with valuable features that will encourage more teachers to use the system. To achieve this goal, the paper explores the use of a headless Content Management System (CMS) called Strapi. The primary research question the paper seeks to answer is whether a headless CMS, specifically Strapi, can provide a good UX to teachers. A usability evaluation of the built-in Strapi UI for content creation and management reveals it to be generally consistent and user-friendly but challenging and unintuitive to create courses with programming exercises. As a result, the decision was made to develop a new teacher’s UI based on the existing Agni UI for students in an editable version. Once the development is complete, a new usability evaluation of the fully developed teacher’s UI will be conducted with the Strapi UI evaluation as a baseline for comparison.

**2012 ACM Subject Classification** Applied computing → Interactive learning environments

**Keywords and phrases** learning environment, programming exercises, programming learning, automatic assessment, headless CMS, CMS, user experience

**Digital Object Identifier** 10.4230/OASICS.ICPEEC.2023.4

**Category** Short Paper

**Funding** This research was conducted within the “FGPE Plus: Learning tools interoperability for gamified programming education” project supported by the European Union’s Erasmus Plus programme (agreement no. 2020-1-PL01-KA226-HE-095786), and financed by National Funds through the Portuguese funding agency, FCT – Fundação para a Ciência e a Tecnologia, within project LA/P/0063/2020.

## 1 Introduction

Learning programming can be challenging for beginners. Winslow [13] noted that many novice programmers might know the syntax and semantics of individual statements but struggle with combining these features into valid programs. His and other studies [11, 3] have emphasized the importance of practice through exercises, which is most effective with immediate feedback. However, it is impossible for teachers to manually provide immediate feedback for every exercise. Automated assessment systems for programming exercises emerged as a solution, freeing teachers to focus on students needing additional support.



© Yannik Bauer, José Paulo Leal, and Ricardo Queirós;  
licensed under Creative Commons License CC-BY 4.0

4th International Computer Programming Education Conference (ICPEEC 2023).

Editors: Ricardo Alexandre Peixoto de Queirós and Mário Paulo Teixeira Pinto; Article No. 4; pp. 4:1–4:8

OpenAccess Series in Informatics



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

## 4:2 Can a CMS Provide a Good User Experience to Teachers?

While there are e-learning systems that offer the ability to create and manage courses with automated assessment, many of them were designed to focus on the UX of students and less on the teacher's UX, which can result in fewer teachers utilizing them.

This paper presents the design and development of a UI to manage and author course contents with automated assessment for Agni, a code playground for learning JavaScript. Previously, Agni consisted only of a UI for students. The course contents were saved in configuration files without the ability to alter or create them through an interface. The main objectives for the UI are 1) a good user experience for teachers; 2) an effective way of creating, adapting, and managing course contents with automated assessment; 3) a repository of exercises with the possibility of extending it with other repositories and 4) allowing for sequencing the course content.

The idea was to use a headless CMS to achieve the goal. This offers the flexibility to use a separate UI for displaying the content (in our case, the Agni Student UI) without being tied to a specific model, as with traditional CMSs. Also, headless CMSs provide a UI to manage and create the contents, which the teachers could use. Strapi was selected for this task. However, a usability evaluation showed that the UI for creating and managing content was generally consistent and user-friendly but challenging and unintuitive for creating course content with programming exercises. This, and the inability to customize the UI, led to the decision to develop a new UI for the teachers. The design approach was to leverage the Agni student's UI and make it editable. This means having input fields instead of text boxes, for example, for lesson names, and adding missing functionalities, such as the addition of new modules. The work is ongoing, and once development is complete, a usability evaluation of the fully developed teacher's UI will be conducted and assessed its effectiveness with the previous evaluation as a benchmark.

The remainder of this paper is organized as follows. Section 2 presents the state of the art, covering user experience and existing programming virtual learning systems (VLEs). The following section describes the main parts of the user interface, a presentation of the data model, and a subsection about the API. The evaluation method and evaluation of Strapi's UI are presented in Section 4. The final section summarizes the paper's contributions and highlights future work.

## 2 State of The Art

The field of programming VLEs is constantly evolving, with emerging trends and innovations shaping how students interact, and teachers create programming courses. This section starts with a general review of UX and explores teachers' UX in VLEs, analyzing their features and missing potentials.

### 2.1 User Experience

According to ISO 0241 – 210, the definition of UX is “A person's perceptions and responses that result from the use or anticipated use of a product, system or service.”. However, the concept of UX encompasses various aspects, leading to multiple possible definitions, dynamic concepts, and theoretical models, including aesthetics, usability, effectiveness, pleasure-based, and emotional experience. [4, 1, 6] provide more detailed information about the different concepts and challenges related to UX.

When designing a UI, authors agree that the focus must be on the user's needs rather than solely relying on the designer's perspective. Therefore, evaluation is crucial to gain insights into the user's experience. There are several evaluation methods, including questionnaire-based evaluation, such as the User Experience Questionnaire (UEQ) by Martin Schrepp,

Bettina Laugwith, and Theo Held [5], or a questionnaire based on the 10 usability heuristics for User Interface Design described by Jakob Nielsen in [7]. These heuristics include visibility of system status, match between system and real world, user control and freedom, consistency and standards, error prevention, recognition rather than recall, flexibility and efficiency of use, aesthetic and minimalist design, help users recognize/diagnose and recover from errors, and help and documentation. Other evaluation methods are explained in more detail in [2, 8]

## 2.2 System Review

Various systems are available to assist teachers in creating and managing programming courses, including Udemy, Mooshak 2, and Moodle. Typically, the course content is structured within one or two levels, consisting of modules and lessons where teachers can create materials and exercises.

The most common exercises with automated assessment are multiple-choice quizzes and programming exercises, supported by many systems such as Udemy, Mooshak 2, and Coderbyte. The correctness of the student's code for programming exercises is being evaluated dynamically. This means that predefined test cases run on the code to verify its correctness and efficiency and give immediate feedback to the student. The tests can be divided into dynamic and static testing [12]. Static testing refers to tests for which the code does not have to run, for example, checking for expression usage or code design. On the other hand, dynamic testing is performed on code execution. Unit tests are an example of dynamic tests. Most e-learning systems with automated assessment only support the creation of dynamic tests that can either be introduced with an input and wanted output field or a file for the teacher to write the whole code for unit tests. Other helpful features, such as an exercise repository, are often missing [10].

Sequencing the course content is another interesting feature. It can be based on time conditions, where the teacher can define a date or number of weeks/days after which the students can work on specific content. Alternatively, content can be sequenced based on the progress of exercises. Mooshak 2, for example, supports the sequencing of all content based on time and exercises completed.

Although these functionalities are important, they lose their impact with an unintuitive UI. The majority of systems involve a strategy of “form filling”, where the interface consists of pairs of field names and inputs to declare the data of the contents. Some systems, such as Udemy, support the visualization of the created content in a student's view to see how it will appear. Moodle's teacher UI is more similar to an editable version of the Student UI with additional functionalities, such as adding lessons and declaring metadata.

## 3 System Design

Agni is a web-based code playground designed for learning JavaScript and providing students with a user-friendly interface for programming courses with exercises that are being evaluated automatically. However, a crucial feature was missing - a dedicated interface for teachers to create, manage, and customize the course content, along with a backend system to independently store them. Previously, the content was saved in configuration files with no interface for editing or creation.

To address this issue, Strapi, a headless CMS, was chosen as the solution for content management. Strapi offers both a user-friendly interface and an API for quickly creating a data model and managing/creating content. The API supports CRUD (Create, Read, Update, Delete) operations and can be customized using hooks. Strapi provides three types

## 4:4 Can a CMS Provide a Good User Experience to Teachers?

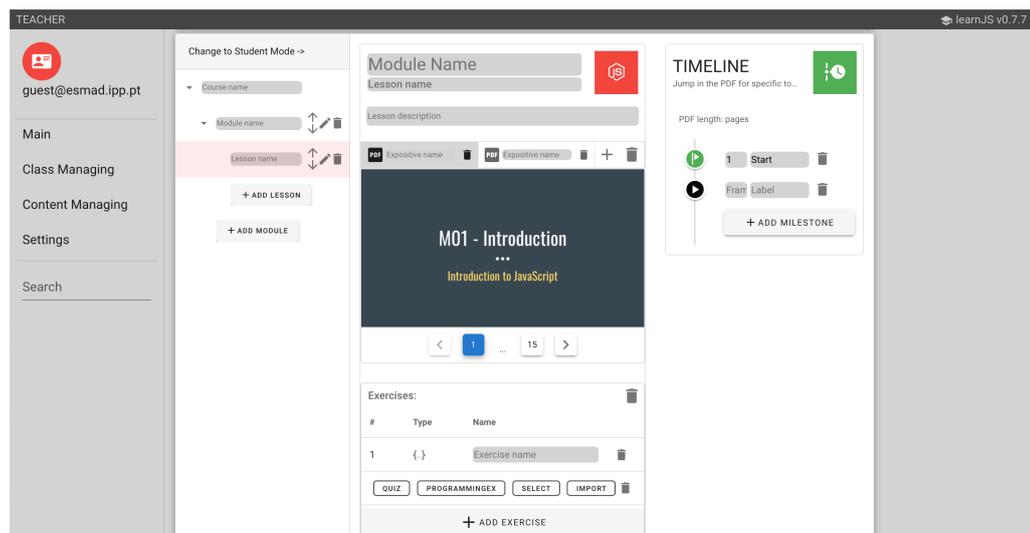
of data structures: single types, collection types, and components. Single types and collection types have an API endpoint and can be created and edited independently. Components are reusable structures that can be used in different collections and single types.

While Strapi's UI for content management was initially considered a hypothesis for teachers to create and manage content, a usability evaluation (explained more in Section 4) revealed that it was generally consistent and user-friendly but challenging and unintuitive for creating course content with programming exercises. Additionally, the lack of customization options in the UI led to the decision to develop a new UI to ensure a better UX for the teachers. The design of this UI, the data model to support it, and the API for the communication between the UI and the server will be explained in the following subsections.

### 3.1 User Interface

The teacher's UI consists of two main parts: class management and content management. The former is to import, create, and manage students and their work, which is not fully developed yet, and the latter is to create and modify courses with their materials. To ensure a positive UX for teachers and provide an intuitive interface for creating and editing courses, the Agni student's UI was transformed into an editable version (see Figure 1). This approach enables teachers to see exactly what they are modifying and how it will look to students. Currently, the system only supports course creation.

Figure 1 displays a screenshot of the UI in the state of editing or creating a course. It is divided into three panels. The left panel is a navigation menu. The right panel shows buttons for actions on the middle panel, such as save the course or exit. The middle part is the main panel where teachers are able to see the different courses, classes, etc., and edit them. In the main panel of Figure 1, the editable Agni student interface for creating or editing courses can be seen. In order to make the student UI editable, text fields, such as module name and lesson description, were transformed into input fields. Additionally, buttons add functionalities, such as creating or deleting lessons, exercises, and other elements. Furthermore, the edit icons in the menu open dialog boxes to introduce information, such as conditions for when a student can work on a lesson that the student cannot see directly in the UI.



■ **Figure 1** UI while editing or creating a course.

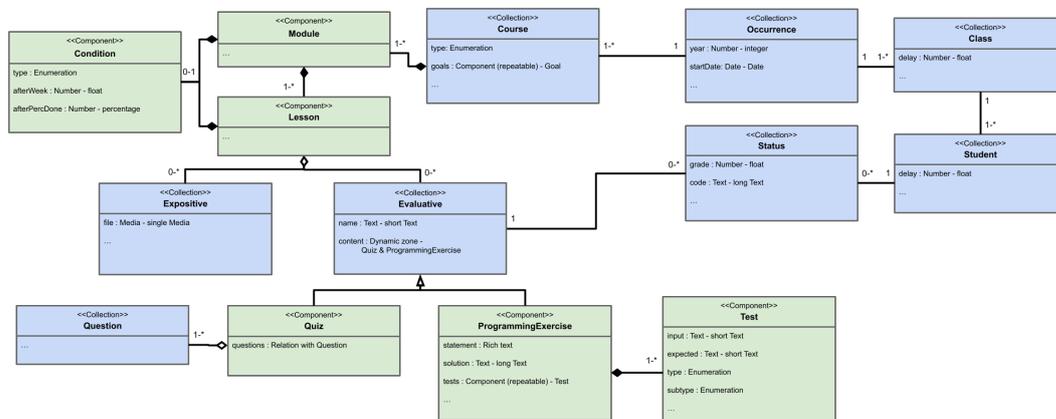
### 3.2 Data Model

The data model for the content created by teachers and viewed by students is illustrated in Figure 2. It has been implemented in Strapi and can be divided into two substructures: one for managing the classes, which teachers can access in the class management navigation, and the other for managing courses and their contents, which teachers can access in the content management navigation.

A *Course* is composed of *Modules*, which contain *Lessons*, that can include multiple *Expositive* and *Evaluative* contents. *Expositives* are used to present information to students and can be in the form of PDF files or video files. *Evaluatives*, on the other hand, are exercises that can be in the form of multiple-choice quizzes or programming exercises. To automatically evaluate these programming exercises, teachers can create *Tests* with input, expected output, type (log, expression, metric, function), and subtype (error for expression, occurrences, and lines for metric) parameters. This provides teachers with a quick way to create different types of tests without having to write complex code for unit tests. In order to sequence *Modules* and *Lessons*, they contain conditions with fields named “afterWeek” to specify after which week the student can work on a *Module* or *Lesson*, and “afterPercDone” to define a percentage of completed exercises after which the student can progress.

*Occurrences*, *Classes*, *Students*, and *Statuses*, form the structure for managing classes. *Occurrences* are linked to a *Course* and have a start date, determining when students can start accessing the *Course*. The “afterWeek” condition for *Modules* and *Lessons* is based on this start date. *Classes* and *Students* have fields for declaring delays that may occur during the year. The grade and solution to a programming exercise done by the student are saved in the *Status*.

*Course*, *Expositive*, *Evaluative*, and *Question* were chosen as collection types for easy reuse and individual editing. Similarly, *Occurrence*, *Class*, *Student*, and *Status* were chosen as collection types for quick and convenient access and creation.



■ Figure 2 Main parts of the Data Model.

### 3.3 API

Strapi offers a RESTful API as the primary means for communication between UI and server. The API supports the full range of CRUD operations, including creating, reading, updating, and deleting content within collection types. These operations can be customized to the

## 4:6 Can a CMS Provide a Good User Experience to Teachers?

specific needs of individual collection types through hooks. Additionally, the API supports a wide range of query parameters, which can be utilized to filter, sort, and paginate data within requests.

To optimize performance and reduce communication between the UI and server, the possibility of creating multiple instances of collection types with one request was implemented. Furthermore, the creation of a complete course structure encompassing different collection types, such as *Course*, *Evaluatives*, *Expositives*, and *Questions*, with one request was developed.

The API is secured using Strapis' built-in user-permission plugin, in which user roles were created for students and teachers. This plugin allows the customization of permissions for each collection and request type, depending on the user's role. For instance, only teachers are authorized to create content. When teachers create content, the author's identity is also saved in the data structure. This enables the declaration of permissions for modifying or getting content created by other teachers. For example, can they only access the data of their students. As for the students, the API enables them to access only their course, with the modules and lessons viewable contingent on the conditions established by the teacher.

### 4 Validation

The evaluation is an essential part of creating a UI with a good UX. Strapi's UI, which was the first idea for the teachers, was evaluated using a satisfaction questionnaire. Since the result was not satisfactory, the decision was made to develop a new UI for the teacher. The work is ongoing, and after finishing it, a satisfaction evaluation will be done with the previous evaluation used as a baseline for comparison and evaluating progress. The evaluation methodology, results of the Strapi UI evaluation, and its conclusion will be presented next.

#### 4.1 Evaluation Methodology

The evaluation was done using a satisfaction questionnaire based on the ten usability heuristics for User Interface Design from Jakob Nielsen [7]. These are visibility of system status, match between the system and the real world, user control and freedom, consistency and standards, error prevention, recognition rather than recall, flexibility and efficiency of use, aesthetic and minimalist design, help users recognize/diagnose and recover from errors, and help and documentation. These ten heuristics and additional the easiness of learning, speed, and reliability of functions and tasks that the system wants to solve were evaluated with multiple questions on a five-point Likert scale (1 - Never, 2 - Almost Never, 3 - Regular, 4 - Almost always, 5 - Always). The questionnaire also included an overall classification of the system on a five-point scale (1 - Bad, 2 - Insufficient, 3 - Sufficient, 4 - Good, 5 - Very Good) and text fields to describe strong points, weak points, and improvement suggestions. Before completing the questionnaire, the respondents were required to carry out typical tasks performed by a professor, such as creating courses, reusing course materials, and associating them with students. Following Jakob Nielsen's proposal [9], five computer science master students from the University of Porto were selected to evaluate Strapi's UI, as this number is sufficient to identify the majority of usability issues.

#### 4.2 Results Strapi UI

Figure 3 shows the evaluation heuristics with its average mean score and standard deviation of their questions. Many of them leading to a positive spectrum, especially speed, consistency, and emphasis, with a mean score of about 4, were evaluated positively. On the other hand,

flexibility, easiness, reliability, and overall classification, with a medium score of 2, 2.2, 2.9, and 2.8, respectively, were evaluated negatively. As strong points were mentioned, easy creation of a *Course* with *Modules* and *Lessons*, excluding *Expositives* and *Evaluatives*, and also the generally easy-to-use and fast UI. Weak points described were the need to create collection types like *Course*, *Expositive*, *Evaluative*, and *Question* individually and only after that being able to associate them with each other, which is unintuitive and time-consuming. The lack of a help system, a landing page without helpful information, and no explanation of some fields were also critiqued. A tutorial was added as an improvement suggestion.

The overall more positive evaluation of the 10 usability heuristics for Strapi's UI was expected due to it being one of the positive aspects of why people use Strapi. However, the Strapi UI is too generic for the specific necessities of managing programming courses. Different collection types can only be created separately, which makes the creation of a whole course time-consuming and not effective. Also, the inability to create a hierarchy of collection types or define different sizes for fields makes it difficult to organize and use the space in an efficient way. This led to the conclusion that the Strapi UI is not sufficient to achieve the objective of a good UX for the teacher. Nevertheless, the API and database can be used due to the ability to customize using hooks.

While students provided valuable insights into the UX, selecting teachers as respondents for the evaluation would have been more appropriate. Teachers possess a deeper understanding of the specific needs and requirements associated with their role within the system.

Category	Mean	SD
1. Visibility	3.25	0.94
2. Compatibility	3.17	0.82
3. Freedom	3.14	0.73
4. Consistency	3.94	0.87
5. Prevention	3.11	0.48
6. Emphasis	3.93	0.59
7. Flexibility	2	0.80
8. Aesthetics	3.25	0.43
9. Help to Users	3.2	0.53
10. Help with documentation	3.25	1.04
11. Easiness	2.25	1.12
12. Speed	4.2	0.49
13. Reliability	2.91	1.19
14. Classification	2.8	0.61

■ **Figure 3** Strapi's UI evaluation results.

## 5 Conclusion and Future Work

This paper highlights the advancements made in the development of a UI with a good UX and useful functionalities for teachers to manage and create courses. One of the main contributions of this paper is the examination of whether a headless CMS, specifically Strapi, can adequately serve this purpose. The usability evaluation indicated it to be too generic and not flexible for effectively creating and managing programming courses.

Furthermore, the paper presents a promising approach to leverage the Agni student's UI and make it editable by having input fields instead of text boxes and adding missing functionalities, such as the addition of lessons for the teacher. The UI communicates with the customized API of Strapi to create and manage the contents of the data model.

In future work, features such as the reuse of exercises will be implemented. The possibility to import external contents, especially exercises, as well as the design and implementation of the class managing part, including the import and creation of students, will be finished. After that, a final evaluation of the UI will be conducted using the Strapi UI evaluation as a benchmark for comparison and evaluation of the progress.

---

## References

- 1 Allam Hassan Allam, Ab Razak Che Hussin, and Halina Mohamed Dahlan. User experience: challenges and opportunities. In *Journal of Information Systems Research and Innovation 2013*, 2013.
- 2 D. Benyon. *Designing User Experience*. Pearson Educación, 2019. URL: <https://books.google.pt/books?id=MXqFDwAAQBAJ>.
- 3 John Dunlosky, Katherine A. Rawson, Elizabeth J. Marsh, Mitchell J. Nathan, and Daniel T. Willingham. Improving students' learning with effective learning techniques: Promising directions from cognitive and educational psychology. *Psychological Science in the Public Interest*, 14(1):4–58, 2013. URL: <http://www.jstor.org/stable/23484712>.
- 4 Marc Hassenzahl and Noam Tractinsky. User experience – A research agenda. *Behaviour & Information Technology*, 25(2):91–97, 2006. doi:10.1080/01449290500330331.
- 5 Bettina Laugwitz, Theo Held, and Martin Schrepp. Construction and evaluation of a user experience questionnaire. In *HCI and Usability for Education and Work*, volume 5298, pages 63–76, November 2008. doi:10.1007/978-3-540-89350-9\_6.
- 6 Effie Lai-Chong Law, Virpi Roto, Marc Hassenzahl, Arnold P.O.S. Vermeeren, and Joke Kort. Understanding, scoping and defining user experience: A survey approach. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 719–728, New York, NY, USA, 2009. Association for Computing Machinery. doi:10.1145/1518701.1518813.
- 7 Jakob Nielsen. Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, pages 152–158, New York, NY, USA, 1994. Association for Computing Machinery. doi:10.1145/191666.191729.
- 8 Jakob Nielsen. Usability inspection methods. In *Conference Companion on Human Factors in Computing Systems*, CHI '94, pages 413–414, New York, NY, USA, 1994. Association for Computing Machinery. doi:10.1145/259963.260531.
- 9 Jakob Nielsen. Why you only need to test with 5 users, March 2000. URL: <https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/>.
- 10 Ricardo Queiros and José Leal. Programming exercises evaluation systems – An interoperability survey. In *International Conference on Computer Supported Education*, volume 1, pages 83–90, January 2012.
- 11 Roshni Sabarinath and Choon Lang Gwendoline Quek. A case study investigating programming students' peer review of codes and their perceptions of the online learning environment. *Education and Information Technologies*, 25(5):3553–3575, September 2020. doi:10.1007/s10639-020-10111-9.
- 12 Zarina Shukur, Edmund Burke, and Eric Foxley. The automatic assessment of formal specification coursework. *Journal of Computing in Higher Education*, 11(1):86, 1999.
- 13 Leon E. Winslow. Programming pedagogy – A psychological overview. *SIGCSE Bull.*, 28(3):17–22, September 1996. doi:10.1145/234867.234872.

# Sifu Reloaded: An Open-Source Gamified Web-Based CyberSecurity Awareness Platform

José Carlos Paiva  

CRACS – INESC TEC, Porto, Portugal  
DCC – FCUP, Porto, Portugal

Ricardo Queirós   

CRACS – INESC TEC, Porto, Portugal  
uniMAD – ESMAD, Polytechnic of Porto, Portugal

Tiago Gasiba  

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

---

## Abstract

Malicious actors can cause severe damage by exploiting software vulnerabilities. In industrial settings, where critical infrastructures rely on software, handling these vulnerabilities with utmost care is crucial to prevent catastrophic consequences. For this purpose, a cybersecurity awareness platform called Sifu was created. This platform automatically assesses challenges to verify its compliance to secure coding guidelines. Using an artificial intelligence method, an interactive component provides players with solution-guiding hints. This paper presents an improved version of the Sifu platform, which evolves the tool in the following aspects: architecture, data model and user interface. The new platform separates the server and client-side using a REST API architecture. It also accommodates an intrinsic and richer layer of gamification, which explores the concept of game rooms at an organizational and gamification level. Finally, it offers an improved interactive training experience for individuals and organizations through a responsive and intuitive single-page web application.

**2012 ACM Subject Classification** Applied computing → Interactive learning environments; Applied computing → E-learning; Applied computing → Computer-managed instruction; Applied computing → Computer-assisted instruction; Security and privacy

**Keywords and phrases** learning environment, cybersecurity, challenges, gamification, automatic assessment

**Digital Object Identifier** 10.4230/OASICS.ICPEEC.2023.5

**Category** Short Paper

**Funding** This work is financed by National Funds through the Portuguese funding agency, FCT – Fundação para a Ciência e a Tecnologia, within project LA/P/0063/2020.

## 1 Introduction

In an increasingly interconnected world, the importance of cybersecurity awareness cannot be overstated. The rapid proliferation of digital technologies and increased cyber threats demand proactive measures to safeguard critical infrastructures. As malicious actors continue to exploit software vulnerabilities, there is a need for practical and effective cybersecurity awareness training that gives individuals and organizations the knowledge and skills to defend against cyber threats. In [3], the Sifu platform was introduced. This platform aims to increase cybersecurity awareness of software developers on secure coding. The Sifu platform, embedded in cybersecurity challenge (CSC) events, is a successful artifact among industrial software developers. CyberSecurity Challenges is a game that targets software developers and was inspired in the Capture-the-Flag (CTF) type of game. In CTF games, the participants



© José Carlos Paiva, Ricardo Queirós, and Tiago Gasiba;  
licensed under Creative Commons License CC-BY 4.0

4th International Computer Programming Education Conference (ICPEEC 2023).

Editors: Ricardo Alexandre Peixoto de Queirós and Mário Paulo Teixeira Pinto; Article No. 5; pp. 5:1–5:8

OpenAccess Series in Informatics



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

are presented with challenges related to cybersecurity in a competitive environment. Upon solving the challenges, the players earn points. The player or team with the most points wins the game.

However, the Sifu platform needs a consolidated gamification aspect and cannot easily be used as a stand-alone application (i.e., separated from CSC events). An improved Sifu platform was developed to address these issues, which we call SifuV2. The new platform profits from the authors' long experience with the original Sifu platform, players' feedback on user experience, and the author's experience in designing gamified applications. The newly evolved platform can empower individuals and organizations in the battle against cyber risks.

This paper aims to introduce the new features and capabilities of this web-based cybersecurity awareness platform based on architecture, data model, and user interface. In the architecture facet, this new version separates client and server through an API complying with REST (Representational State Transfer). The importance of REST architectures lies in their ability to provide scalable, interoperable, and flexible systems that can easily integrate with other services. By adhering to REST principles, developers can build robust, distributed, and easily maintainable software systems that meet the demands of modern web and mobile applications. The second facet is the data model, which adds a new layer of gamification, exploring the concept of game rooms both from an organizational perspective and to gamify the environment. Including gamification elements, such as quizzes, challenges, and rewards, enhances user engagement and motivation, transforming cybersecurity education into an enjoyable and immersive experience.

Additionally, the platform was changed to provide an improved user experience through an intuitive user interface. Here, learners can access a rich array of interactive training modules and engaging videos through a responsive single-page web application.

In the subsequent sections of this paper, we will explore related work, in particular, other web-based cybersecurity awareness platforms. In section 3, we present the evolved platform's architecture, and enumerate the data model changes regarding the previous version. Additionally, we introduce the new user interface. In the last section, the contributions of this article to the scientific community are presented, and future work is briefly discussed.

## 2 Related Work

Table 1 and Table 2 depict currently available cybersecurity awareness training platforms. These platforms typically offer a combination of training modules, phishing simulations, knowledge assessments, and reporting capabilities to help organizations educate their employees about cybersecurity threats and promote a culture of security awareness. Among these, we can find the following training platforms:

- **Cybrary**<sup>1</sup>: offers a wide range of cybersecurity courses, which covers various topics such as network security, ethical hacking, and incident response. This platform is offered as a free or as a paid version. The platform includes the option to earn cybersecurity certifications.
- **Hack The Box**<sup>2</sup>: a platform offering a range of realistic virtual machines for penetration testing practice. It provides challenges and labs allowing participants to test their skills in a controlled environment. Access to some machines and features requires a paid subscription, but a free option is also available.

---

<sup>1</sup> <https://www.cybrary.it>

<sup>2</sup> <https://www.hackthebox.eu>

- **TryHackMe**<sup>3</sup>: is a platform that teaches cybersecurity through interactive and gamified content. It offers various rooms with different themes and difficulty levels. Each room includes practical exercises, challenges, and walkthroughs to help players learn and practice cybersecurity concepts.
- **PentesterLab**<sup>4</sup>: provides online courses and hands-on labs specifically designed for web application security. Their courses cover web penetration testing, network security, and secure coding. They offer a mix of free and paid content, with interactive labs for skill practicing.
- **OverTheWire**<sup>5</sup>: is a platform that offers a series of interactive war games designed to teach and practice cybersecurity skills. Each game focuses on a specific area of cybersecurity, such as e.g. basic Linux command-line skills and cryptography. The challenges' difficulty gradually increase, allowing the player to improve their skills progressively.
- **Root-me**<sup>6</sup>: is a platform that offers a wide range of challenges and labs covering various cybersecurity topics, including web security, network security, reverse engineering, and more. It provides a hands-on approach with practical activities to reinforce players' understanding and skills.
- **CTFtime**<sup>7</sup>: while not a training platform, this website aggregates information about existing Capture The Flag (CTF) competitions that take place worldwide.

Evaluating each platform based on the organization's needs and requirements is essential before deciding which to use. In [4], the authors provide guidelines on selecting CTF games to address industrial software developers. Employing a set of well-defined criteria facilitates the comparison of tools, allowing one to make an informed decision and choose the most suitable web-based platform for specific cybersecurity needs. It also provides a structured approach to assess the strengths and weaknesses of each tool, making it easier to identify the best fit for any specific use case.

Predefined criteria were selected and applied to compare cybersecurity tools effectively. The results are presented in two tables according to their pricing models: Table 1 with those that offer both free and paid models (or exclusively paid) and Table 2 with the free tools.

■ **Table 1** Comparison of paid cybersecurity tools.

Criteria	Cybrary	Hack The Box	TryHackMe	PentesterLab
Year	2015	2017	2018	2013
OS/Proprietary	Proprietary	Online Platform	Online Platform	Online Platform
User Experience	✓	✓	✓	✓
Course Catalog	✓	✓	✓	✓
Course Topics	Various	Ethical Hacking	Ethical Hacking	Penetration Testing
Challenge Types	N/A	CTF	CTF	N/A
Certification	✓	×	×	×
Gamification	✓	✓	✓	×
Automatic Eval.	×	×	×	×
Collaboration	✓	×	✓	✓

<sup>3</sup> <https://tryhackme.com>

<sup>4</sup> <https://pentesterlab.com>

<sup>5</sup> <https://overthewire.org>

<sup>6</sup> <https://www.root-me.org>

<sup>7</sup> <https://ctftime.org>

■ **Table 2** Comparison of free cybersecurity tools.

Criteria	OverTheWire	Root-me	CTFtime
Year	2002	2004	2008
OS/Proprietary	Online Platform	Online Platform	N/A
User Experience	✓	✓	✓
Course Catalog	✓	✓	✓
Course Topics	Networking/Security	Challenges/CTF	Challenges/CTF
Challenge Types	Challenges/CTF	Challenges/CTF	Challenges/CTF
Certification	×	×	×
Gamification	×	×	×
Automatic Eval.	×	×	×
Collaboration	×	✓	✓

Since currently available tools have different pricing models, ranging from free to paid options, this criteria allows users to choose a platform that fits their budget and financial constraints.

All the covered platforms offer a comprehensive course catalog covering various aspects of cybersecurity. This indicates that there are abundant learning opportunities available across different platforms. The tools also cover various course topics, including ethical hacking, penetration testing, networking/security, and challenges/capture the flag. While there is a large variety in the subjects offered, a quality assessment of these is outside the scope of the present work. We refer the reader to [4] for further discussions.

Cybrary is the only platform that offers certifications upon completion of their courses. This is beneficial for individuals looking to enhance their credentials and showcase their skills in the cybersecurity field.

Regarding gamification, Hack The Box, Cybrary, and TryHackMe incorporate several gamification features which can enhance engagement and motivation. The former offers simulations with avatars and narrative scenarios to throw users into classic cyber hacks. The others include gamification features typically found in PBL (Points/Badges/Leaderboards) gamified applications. In terms of collaboration, Cybrary, TryHackMe, Root-me, and CTFtime have collaborative challenges/missions that require teams to work together to accomplish a common goal. In this realm, most tools also offer social features, such as discussion chats and forums, which allows participants to share ideas and build relationships, enhancing collaboration and promoting a sense of community.

Finally, as indicated by the checkmarks, most tools strive to provide a positive user experience. This suggests that they are designed considering usability and accessibility, making it easier for learners to navigate and engage with the platforms.

### 3 SifuV2

Sifu [3] is an open-source web-based platform for cybersecurity awareness that challenges users to fix the source code of a project containing vulnerabilities. User submissions are automatically accepted if the source code complies with secure coding guidelines, does not have known or additional vulnerabilities, and performs the desired tasks. To this end, the analysis of the proposed solution submitted by a user involves several tools (e.g., SonarQube [9], CodeChecker [2], and various unit testing tools) acting on different phases of the analysis, static and dynamic. An intelligent coach processes the result of such analysis to complement the result presented to the learner with a hint.

SifuV2 keeps both the original automated assessment module and the intelligent coach module, which are the central components of Sifu. However, it reformulates the server and client sides. In particular, it separates both by adopting a REST (Representational State Transfer) API architecture on the server side. The REST API is then connected to the front end by a single-page application (client). The REST API is implemented using Flask [8] and is supported with a PostgreSQL database. The following subsections present the significant changes present in the second version of Sifu, which concentrate on the data model and user interface.

### 3.1 Data Model

Sifu v2 aims to redesign the previous version [3] to accommodate an intrinsic and richer layer of gamification rather than only awarding points for correctly solving a challenge presented in an external leaderboard and giving the platform a more formative direction. To this end, it explores the concept of game rooms from an organizational and gameful perspective. Firstly, rooms represent checkpoints of the learning path a player must complete before proceeding to another. Lastly, rooms display the active players solving the challenges simultaneously, track the best performers using a leaderboard, and reward players on successful completion.

Consequently, the data model has been re-implemented to achieve these goals. The most important model components are the following.

**User** represents both administrators and players. Each player has experience points, levels, badges, medals, and a wallet containing coins. Moreover, they have a public profile containing the username, avatar, level, medals, and badges.

**Room** has a set of generally related challenges to be solved (e.g., the same vulnerability or project). Platform administrators create rooms that can be either of the type: “open”, “accessed by invite”, or “locked by password/access code”. Completing a room gives the player one or more rewards, possibly unlocking another room directly (i.e., providing an access code) or indirectly (i.e., achieving a level of experience that enables access to the room). Each room has an internal leaderboard sorted in descending order by the number of solved challenges and, in case of tie, in ascending order of shortest solving time. Furthermore, players can also create duel rooms by using coins. These rooms are open to a limited amount of players, who should pay an established entry fee and run until one of the participants finalizes all challenges. Such participant collects the sum of coins paid by all participants.

**Challenge** is the educational unit of Sifu. The traditional Sifu challenges are projects that contain one or more vulnerabilities, which ask the player to rewrite the code keeping the same functionality but eliminating existing vulnerabilities and following secure coding guidelines and secure software development best practices. Quizzes are the other type of challenge supported in Sifu. Challenges contain all information and necessary files for the presentation of the task to the player as well as its evaluation by the automated assessment module and hint generation by the artificial intelligence coaching module. Although challenges can only be accessed through rooms, a pre-association to a room is not mandatory. Challenges left disconnected are randomly selected for duel rooms when these are started. Any challenge can reward the player on completion, independently of the room.

**Hint** belongs to a challenge and defines the name, description, and cost (in coins) of the hints generated by the intelligent coach, according to its priority in the ranking of secure coding guidelines, as presented in Gasiba et al. [5]. Actual hints are generated on players' request, and they can opt to buy or not the hint.

**Leaderboard** holds the name, sorting metric (e.g., points and coins), context (global or room-scoped), and reset interval of the leaderboards. Being on top-3 of a leaderboard awards the player a medal at each reset.

**Reward** is a virtual item given to a player for completing a room or challenge or leveling up. SifuV2 includes four types of rewards, namely coins, experience points, badges, and access codes. For the first two, an amount should be set.

**Question and Answer** enable players to submit clarification requests about challenges and administrators to answer them, respectively.

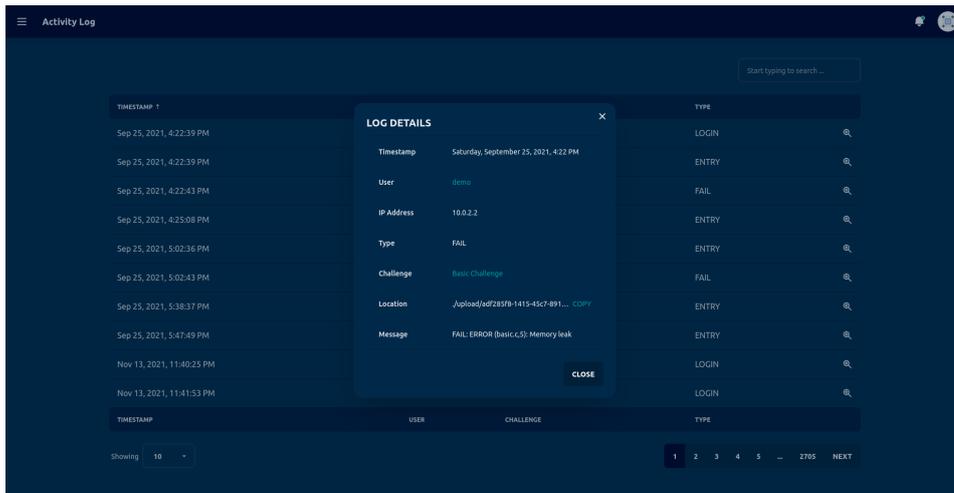
### 3.2 User Interface

The user interface (UI) is a single-page web application developed in Vue 3 with TypeScript, using Tailwind CSS [6] – a utility-first CSS framework packed with several classes that can be composed to build any design directly from the markup. An additional free and open-source Tailwind CSS component library – daisyUI [10] – has been applied to obtain a more characteristic design line. The UI colors, with dark and light themes, follow the company’s brand guidelines. Non-authenticated users share the views for registration, email verification, request a password reset, reset password, and signing in. These pages are identical, containing a dojo background, a light/dark theme switch on the screen’s bottom-left, and a centered card with the corresponding form. Authentication/authorization is based on access and refresh JSON Web Tokens (JWTs); the former is part of the managed state, whereas the latter is stored in local storage. The global state is managed through Vuex [1], a state management pattern. The library serves as a centralized store for all the components in an application, with rules ensuring that the state can only be mutated predictably.

Logged-in users are either administrators or players, each having its separate SifuV2 environment. The administration side works similarly to a content management system and, thus, has much in common with the characteristic UIs of those systems. A top navigation bar with a togglable menu on the left, the notifications icon, and the user avatar on the right. The content area has a paginated data table with an associated search box (right) on resource listing views, while the details view has the associated resource form. The administrators set up all the content available to the players, including the creation of the rooms, the association of challenges to them, and the definition of rewards delivered on the successful completion of such challenges. Moreover, they are also able to consult the submissions, profiles, leaderboards, and all player activity, as well as to clarify doubts sent by players about a challenge. Figure 1 shows the Activity Log view (i.e., listing view of the events of Sifu v2 users) after clicking the zoom-in button to check the details of an evaluated submission.

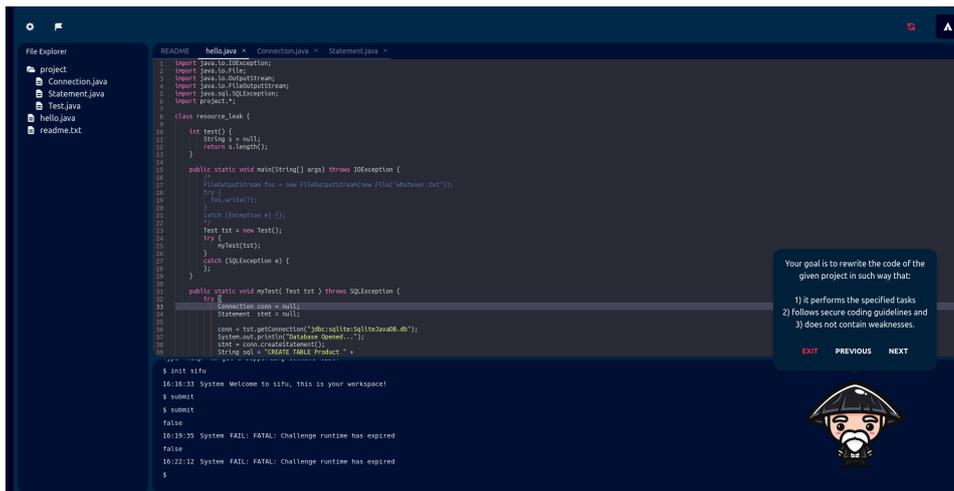
The player’s dashboard consists of a grid of cards corresponding to rooms. Each card contains the room name and description, an indication of the status relative to the authenticated user, and deck-like visualization of the avatars of at most three randomly selected users who are active in the room (with an indication of the total active users in the room). The player can search a specific room using the search box and hide/show open, closed, completed, and locked rooms. One distinctive aspect of the player environment is that the navigation bar is located on the left rather than on top. This navigation bar can be collapsed to prevent distraction and give extra space.

The Challenge Solving view in Figure 2 has the look and feel of a traditional (simple) code editing application. On the left, the workspace files are presented as a file tree. On the bottom is a simple console emulator where the learner can run several predefined commands and see their outcome (e.g., `submit` to send code for evaluation). The content area has a code



■ **Figure 1** Activity Log view on the administrator panel, with details of an evaluated submission opened in a modal box.

editor based on Monaco Editor [7], the editor that powers Visual Studio Code, displaying the contents of the active file and tabs for other opened files. On the top of the screen are buttons (from left to right) to change editor settings, report the challenge, reset workspace files, and submit the current files for evaluation. A vital component of this view is the Sifu Master. This animated avatar interacts with the player, providing guidance through hints (when explicitly requested) or presenting the UI tour. The avatar is a CSS3 animated SVG, which can be moved to any position on the screen, with several animations executed depending on the action performed, including looking right/left, blinking right/left eye, waving right/left arm, boiling hat, talk, panic, greeting with the arm, celebrate, angry, and wind on beard and mustache.



■ **Figure 2** Challenge Solving view on the players environment.

## 4 Conclusion

Cybersecurity awareness is more and more inevitable in this digital world. The need for professionals with the required knowledge and skills to defend against cyber threats is a top priority in big companies and any digitalized company.

This paper presents the second version of Sifu, a re-design of a cybersecurity open-source system with automated assessment and coaching into a gamified web-based cybersecurity training platform. In addition to gamifying the platform, SifuV2 implements the server-side to adhere to more scalable, interoperable, and flexible web development standards and a new user interface with a cleaner, more attractive, interactive, and independent design. The changes to the interface reflect the author's experience in the field and the design of gamified platforms.

We expect to conduct an online open experiment to evaluate this platform's usability and learning effectiveness in future work. These aim to demonstrate that this platform can improve the cybersecurity awareness of participants while having a smooth and engaging experience.

---

### References

- 1 Vue JS Community. What is Vuex? – Vuex, 2023. Accessed on May 2nd, 2023. URL: <https://vuex.vuejs.org>.
- 2 Ericsson. CodeChecker, 2023. Accessed on May 2nd, 2023. URL: <https://codechecker.readthedocs.io>.
- 3 Tiago Espinha Gasiba, Ulrike Lechner, and Maria Pinto-Albuquerque. Sifu - a cybersecurity awareness platform with challenge assessment and intelligent coach. *Cybersecurity*, 3(1):24, December 2020. doi:10.1186/s42400-020-00064-4.
- 4 T. Espinha Gasiba, K. Beckers, S. Suppan, and F. Rezabek. On the requirements for serious games geared towards software developers in the industry. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*, pages 286–296, Los Alamitos, CA, USA, September 2019. IEEE Computer Society. doi:10.1109/RE.2019.00038.
- 5 Tiago Gasiba, Ulrike Lechner, Jorge Cuellar, and Alae Zouitni. Ranking Secure Coding Guidelines for Software Developer Awareness Training in the Industry. 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 11:1–11:11, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/OASICs.ICPEC.2020.11.
- 6 Tailwind Labs. Tailwind CSS, 2023. Accessed on May 2nd, 2023. URL: <https://tailwindcss.com>.
- 7 Microsoft. Monaco – The Editor of the Web, 2023. Accessed on May 2nd, 2023. URL: <https://microsoft.github.io/monaco-editor/>.
- 8 Pallets. Welcome to Flask, 2023. Accessed on May 4th, 2023. URL: <https://flask.palletsprojects.com>.
- 9 SonarSource SA. SonarQube, 2023. Accessed on May 2nd, 2023. URL: <https://docs.sonarqube.org/latest>.
- 10 Pouya Saadeghi. daisyUI – Tailwind CSS Components, 2023. Accessed on May 2nd, 2023. URL: <https://daisyui.com>.

# A Gamified Educational Escape Rooms' Framework for Computer Programming Classes

Ricardo Queirós   

School of Media Arts and Design, Polytechnic Institute of Porto, Portugal  
CRACS – INESC TEC, Porto, Portugal

Carla Pinto  

School of Engineering & CMUP, Polytechnic Institute of Porto, Portugal

Mário Cruz  

School of Education & inED, Polytechnic Institute of Porto, Portugal

Daniela Mascarenhas  

School of Education & inED, Polytechnic Institute of Porto, Portugal

---

## Abstract

---

This paper presents a study on the use of gamified educational escape rooms to foster the teaching-learning process of computer programming, based on an user type taxonomy. The ultimate goal of this work is to identify and validate the most suitable gamification elements and mechanics for each user profile, providing case studies that illustrate their implementation. The main contribution of this work is to guide the design process of educational escape rooms in any domain, by considering the needs, preferences, and motivations of different user types.

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

**Keywords and phrases** Escape Room, Gamification, Computer Programming, User Type Taxonomy

**Digital Object Identifier** 10.4230/OASICS.ICPEC.2023.6

**Category** Short Paper

**Funding** This work of M. Cruz and D. Mascarenhas is funded by National Funds through the FCT - Fundação para a Ciência e a Tecnologia, I.P., under the scope of the project UIDB/05198/2020 (Centre for Research and Innovation in Education, inED).

The author C. Pinto was partially supported by CMUP, which is financed by national funds through FCT – Fundação para a Ciência e a Tecnologia, I.P., under the project with reference UIDB/00144/2020.

All authors would like to thank Erasmus + MATH-DIGGER project, ref 2021-1-PT01-KA220-HED-000032234.

## 1 Introduction

Teaching computer programming is a complex task that requires a deep understanding of programming concepts, problem-solving skills, and, as in any other domain, a lot of practice. However, traditional approaches to teaching programming often fail to engage learners, leading to high dropout rates and low learning outcomes [15, 2].

In order to address this issue, new approaches have emerged, such as gamification and educational escape rooms, with the aims to make learning programming more engaging, effective and fun. Gamification involves using game design elements and mechanics in non-game contexts, such as education, to enhance motivation, engagement, and learning outcomes. On the other hand, educational escape rooms are immersive learning environments that simulate real-world scenarios, requiring learners to solve puzzles, complete tasks, and collaborate with others to achieve a common goal [7].



© Ricardo Queirós, Carla Pinto, Mário Cruz, and Daniela Mascarenhas;  
licensed under Creative Commons License CC-BY 4.0

4th International Computer Programming Education Conference (ICPEC 2023).

Editors: Ricardo Alexandre Peixoto de Queirós and Mário Paulo Teixeira Pinto; Article No. 6; pp. 6:1–6:8

OpenAccess Series in Informatics



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

By combining gamification and educational escape rooms, educators can create an interactive and dynamic learning experience that caters to the diverse needs of learners. By using gamification elements and mechanics (e.g. points, badges, leaderboards), educators build a sense of achievement and recognition for learners, while also fostering healthy competition and collaboration. Additionally, educational escape rooms provide an engaging way for learners to apply their programming knowledge in a practical setting, enhancing their problem-solving skills and critical thinking skills.

In this paper, we present a study on the use of gamified educational escape rooms (GEER) to foster the teaching-learning process of computer programming based on the User Type Hexad Taxonomy of Andrzej Marczewski [10]. For each user type profile, a set of gamification elements and mechanics are provided and applied in case studies related with the computer programming domain. The contribution of this work is to guide the design process of educational escape rooms in any domain by considering the needs, expectations, and motivations of different user types.

The rest of the article is structured in three sections: the second section presents related work on educational escape rooms, user type taxonomies and gamification frameworks. The following section present a GEER framework based on the User Type Hexad Taxonomy. Hence, for each user type profile a set of gamification components are assigned and deployed in a specific case study within the realm of the computer programming domain. Finally, the contributions of this article to the scientific community are presented as well as the future work.

## **2 Related work**

This paper aims to present a study on the use of GEER to promote computer programming learning, based on Andrzej Marczewski's Hexad taxonomy of user types [10]. The work seeks to identify the most suitable gamification components for each user profile. With these ideas in mind, this section explores the concepts of educational escape rooms, gamification and user type taxonomies in order to give a richer context to the selected educational approaches, in order to overcome students' difficulties in the computer programming learning.

### **2.1 Gamification**

With the growing demand for computer programming professionals, teaching programming has become increasingly important at different educational levels. Gamification has proven to be a promising approach to make learning programming more attractive for students. Gamification is defined as the use of game elements and mechanics in non-game contexts in order to increase user motivation in several activities [6].

By applying gamification to teaching programming, educators can create a more engaging and challenging environment, making the learning process more interesting. Gamification frameworks, such as Octalysis [4] and Hexad [10], can be used to guide the gamification design process and ensure that the chosen game components are aligned with educational goals and, most important, with the student preferences [13].

In fact, some research stated the effectiveness of gamification in teaching programming. A recent study by Cai et al. [3] presented a systematic review of the literature on gamification in education, showing that gamification can lead to significant improvements in student performance and engagement with content. Furthermore, the use of games and gamification elements in programming education can contribute to the development of cognitive skills, such as critical thinking and problem solving [12].

## 2.2 Educational Escape Rooms

Educational escape rooms are immersive learning environments that simulate real-world situations in which students work in teams to solve problems and complete tasks in a limited amount of time. These escape rooms, typically used for entertainment games, have been adapted for educational purposes as a way to engage students in learning in an interactive and playful way. This approach has received attention from educational researchers, who have investigated how educational escape rooms can be used to promote learning of specific content and develop skills such as teamwork, problem-solving, decision-making, and critical thinking [7].

Some studies have shown that educational escape rooms can be effective in engaging students and enhancing their cognitive and social skills [12]. Moreover, the use of gamification elements in educational escape rooms can further increase student engagement [3].

However, the design and implementation of educational escape rooms can be challenging for educators. Not only for the lack of knowledge on this domain as well the selection of the best tools to do it. Regardless of the tooling set, the most important is that the experience provides students with an interesting and meaningful challenge, which in turn promote the acquisition of specific knowledge and skills in students [9]. It is also necessary that educational escape rooms are carefully planned, so that the proposed tasks and challenges are aligned with the desired learning objectives [14].

## 2.3 User types taxonomies

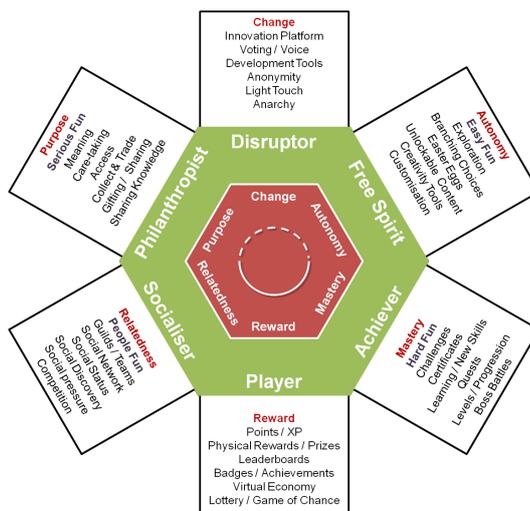
User types taxonomies are frameworks that classify users based on their psychological profiles and behavioral tendencies. These frameworks have been widely used in various fields, including human-computer interaction and marketing, to help designers understand their target audience and create more engaging and effective products [11]. One of the most well-known user type taxonomy is the Hexad framework [10] developed by Andrzej Marczewski, which categorizes users into six types based on their motivations and preferences in software applications:

- **Achievers:** users who seek mastery, completion, and recognition of their accomplishments;
- **Socializers:** users who seek social interaction, collaboration, and feedback from others;
- **Philanthropists:** users who seek to help others, contribute to a cause, and make a positive impact;
- **Free Spirits:** users who seek novelty, freedom, and self-expression;
- **Players:** users who seek competition, challenge, and rewards;
- **Disruptors:** users who seek to break the rules, challenge authority, and explore alternative solutions.

Other popular user type taxonomies include the Bartle Taxonomy of Player Types [1], the Self-Determination Theory of Motivation [5], and the Big Five Personality Traits [8]. These frameworks provide valuable insights into how users interact with software applications and can enrich the design process to create more personalized experiences.

## 3 GEER framework

This section presents a framework for GEER based on the user type Hexad taxonomy (Figure 1). For each identified user profile, we present the most suitable gamification components and apply them in a specific computer programming case study comprising a storyline and a gameplay.



■ Figure 1 User type HEXAD.

### 3.1 Achiever – Cybercrime Investigation

The Achiever is motivated by competition and achievement. They are likely to enjoy an escape room scenario that involves a race against the clock, where they can compete against other teams to see who can escape the room fastest. The room could have multiple paths to escape, allowing the Achiever to choose the most challenging path to prove their skills.

For the achiever user type the most important gamification elements are:

- Challenges and quests to encourage mastery and accomplishment;
- Progress bars and badges to track accomplishments;
- Leaderboards to foster competition and drive motivation.

The case study chosen for this user type is a **Cybercrime Investigation**. In this story, the user is a detective investigating a cybercrime. The suspect has locked the detective in their own office, and the user should escape the room to continue their investigation and catch the suspect.

In terms of gameplay, the room is designed to look like a typical detective’s office, with clues and puzzles hidden throughout the room related to the cybercrime investigation. The puzzles are designed to test the player’s knowledge of computer programming concepts, such as coding languages, data structures, and algorithms. For example, one puzzle could involve decoding a message written in a programming language, while another could require the player to debug a code snippet.

In order to add a competitive element to the achiever profile, the room could have multiple paths to escape, each with increasing levels of difficulty. The user can choose the most challenging path to prove their skills and compete against other users who have completed the room. The room could have a leaderboard displaying the fastest escape times and highest scores, encouraging users to come back and beat their previous records.

### 3.2 Socializer – Hackathon

The socializer is motivated by social interaction and collaboration. They would enjoy an escape room scenario that requires teamwork and communication, with puzzles that can only be solved through cooperation. The room could have a theme that encourages socializing, such as a murder mystery or a haunted house, to make the experience more immersive.

For the socializer user type the most important gamification elements are:

- Collaborative and social challenges that promote teamwork;
- Interactive leaderboards that allow for social comparison and recognition;
- Community features such as forums, chats, and social media integration (likes, following friends and sharing resources).

In order to apply some of these elements the case study will be a **Hackathon**. In this story, the user is a member of a team participating in a hackathon, a collaborative programming competition. The team has been locked in a room and must escape before the time runs out in order to submit their project and win the competition.

The room is designed to look like a typical hackathon workspace, with computer workstations and programming equipment scattered throughout the room. The puzzles are designed to test the player's knowledge of computer programming concepts, such as coding languages, data structures, and algorithms, but also require teamwork and communication to solve. For example, one puzzle could require the player to share their screen with a team member and work together to debug a code snippet, while another could involve coordinating multiple players to input different pieces of code into a shared program. To encourage socializing, the room could also have elements such as a shared whiteboard or a chat system, where players can collaborate and communicate with each other.

Overall, the hackathon theme would make the experience more immersive and exciting, while the collaborative puzzles would promote socializing and teamwork.

### 3.3 Philanthropist – Humanitarian Aid

The philanthropist is motivated by helping others and making a positive impact. They would enjoy an escape room scenario that involves solving puzzles for a good cause, such as finding a cure for a disease or stopping a disaster from occurring. The room could be designed to look like a laboratory or research facility to reinforce the sense of purpose.

For the philanthropist user type the most important gamification elements are:

- Charitable donation or volunteerism opportunities as rewards for achievement;
- Sharing or promoting positive social impact on social media;
- Offering virtual gifts.

In order to apply some of these elements the case study will be a **Humanitarian Aid**. In this story, the user is a member of a team tasked with providing humanitarian aid in a crisis-stricken region. However, the aid shipment has been blocked and the player must escape the room to find a way to get the aid through and help those in need.

In this case study, the room is designed to look like a logistics office, with maps, shipping records, and other aid-related documents scattered throughout the room. The puzzles are designed to test the player's knowledge of computer programming concepts, such as coding languages, data structures, and algorithms, but also require problem-solving and critical thinking skills.

For example, one puzzle could require the player to analyze shipping records and optimize delivery routes using algorithms, while another could involve debugging a program used to track the shipment.

To promote philanthropy, the room could also have elements such as a live feed showing images and videos of the crisis-stricken region, and information about the impact that the aid will have on the local population.

Overall, the humanitarian aid theme would make the experience more emotionally engaging, while the programming puzzles would promote critical thinking and problem-solving skills.

### 3.4 Free Spirit – Virtual Reality Adventure

The Free Spirit is motivated by creativity and self-expression. They would enjoy an escape room scenario that allows them to use their imagination to solve puzzles, with a variety of open-ended challenges that can be tackled in different ways. The room could have a theme that allows for self-expression, such as an art gallery or a music studio.

For the free spirit user type the most important gamification elements are:

- Creative challenges that allow for self-expression and experimentation;
- Non-linear or open-ended gameplay that encourages exploration and discovery;
- Self-guided quests that allow players to set their own goals and direction.

For this type of user the case study selected is related with a **Virtual Reality Adventure**. Here, the user is a virtual reality game developer who has been transported into their own game world by a glitch. The user must escape the game world by solving programming puzzles and fixing the code in order to return to the real world.

The room is designed to look like a virtual reality environment, with futuristic technology and gaming equipment. The puzzles are designed to test the player's knowledge of computer programming concepts, but also require creativity and imagination. For example, one puzzle could involve programming a character to navigate through a maze-like environment, while another could require the player to design and code a new game feature on the spot. To promote free-spiritedness, the room could have elements such as open-ended puzzles that can be solved in multiple ways, or a "sandbox" programming environment where the player can experiment and create without limitations.

Overall, the virtual reality theme would make the experience more exciting, while the programming puzzles would promote problem-solving and critical thinking skills in a fun and imaginative way.

### 3.5 Player – Hacking Adventure

The player user type is motivated by the thrill of the game and the experience of playing. They would enjoy an escape room scenario that is designed like a video game, with a clear objective and challenges that gradually increase in difficulty. The room could have a futuristic or sci-fi theme, with high-tech gadgets and special effects to enhance the gaming experience.

The Player type feels comfortable with the following gamification elements:

- Point systems and rewards for achievements;
- Unlockable content and in-game purchases for added benefits;
- Competitive gameplay with leaderboards and multiplayer options.

In this realm, a **hacking adventure** was the case study selected. The user is a hacker who has been recruited by a secret organization to complete a series of hacking challenges. The user must escape the room by solving programming puzzles and completing hacking tasks in order to prove their skills and earn the organization's trust.

The room is designed to look like a hacker's lair, with computer workstations and hacking tools. The puzzles are designed to test the player's knowledge of computer programming concepts, but also require quick thinking and problem-solving skills. For example, one puzzle could require the player to hack into a computer system using a specific programming

language, while another could involve decoding a complex encryption algorithm. To promote the player's motivation for competition and rewards, the room could have elements such as a leaderboard showing the player's progress and a reward for completing all the challenges.

### 3.6 Disruptor – Cybersecurity Breach

The disruptor user type is motivated by the desire to challenge the status quo and push boundaries. This user type seeks to break down existing barriers and explore new possibilities, often by challenging conventional wisdom and advocating for change.

In this profile, users will be satisfied with the following gamification elements:

- Challenging gameplay that requires strategic thinking and creativity;
- Unique and unconventional gameplay elements that break from traditional game design;
- Risk-taking opportunities, such as high-stakes challenges or timed events.

In this storyline, the user is a cybersecurity expert who has been called in to investigate a major **security breach** at a government agency. The user must escape the room by solving programming puzzles and detecting vulnerabilities in the agency's computer systems in order to prevent further damage and restore security.

The room is designed to look like a government agency's computer center, with advanced technology and equipment related to cybersecurity investigations. The puzzles are designed to test the player's knowledge of computer programming concepts, but also require strategic thinking and creativity. For example, one puzzle could require the user to analyze a system's code and detect any vulnerabilities, while another could involve creating a program to block a hacker's access. To promote disruptive thinking, the room could have elements such as fake news articles or social media posts related to the security breach displayed on the walls. The puzzles could be designed to encourage users to think creatively and outside the box, challenging them to find innovative solutions to the problems they encounter.

## 4 Conclusion

This study has demonstrated the potential of gamified educational escape rooms as a valuable tool for enhancing the teaching-learning process of computer programming. By using a user type taxonomy, the study has identified the motivations, preferences, and needs of different user profiles and provided case studies that illustrate the successful implementation of gamification elements and mechanics for each profile.

The results of this study have practical implications for the design and implementation of educational escape rooms in any domain. By considering the unique characteristics of each user type, designers and educators can create tailored gamification experiences that are engaging, effective, and inclusive. Overall, this paper contributes to the growing body of research on gamification in education and highlights the importance of user-centered design in the development of educational technologies.

---

### References

- 1 Richard A Bartle. Hearts, clubs, diamonds, spades: Players who suit muds. *Journal of MUD research*, 1(1), 1996.
- 2 Ashwin Basawapatna, Ashish Tewari, and Thomas Moher. Identifying and addressing misconceptions about programming: A review of programming knowledge and student errors. *Computer Science Education*, 27(2-3):78–114, 2017.

- 3 Wei Cai, Xiao Ma, and Chin-Lung Lai. Gamification in education: A systematic review. *Journal of Educational Technology Development and Exchange (JETDE)*, 13(1):1–22, 2020.
- 4 Yu-kai Chou. *Actionable gamification: Beyond points, badges, and leaderboards*. Octalysis Media, 2015.
- 5 Edward L Deci and Richard M Ryan. Self-determination theory: A macrotheory of human motivation, development, and health. *Canadian Psychology/Psychologie canadienne*, 49(3):182–185, 2008.
- 6 Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: Defining “Gamification”. In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, pages 9–15. ACM, 2011.
- 7 Fernando Gamboa-Rodríguez, Juan Nicolás García-Sánchez, Eloy García-Sánchez, and Arturo Córdoba-Rangel. Educational Escape Rooms in Higher Education: A Systematic Literature Review. *Education Sciences*, 11(3):114, 2021. doi:10.3390/educsci11030114.
- 8 Lewis R Goldberg. The structure of phenotypic personality traits. *American psychologist*, 48(1):26–34, 1993.
- 9 Chin-Lung Lai, Gwo-Dong Chen, and Rong-Chi Shih. Applying the escape room game approach to course design: A case study of a university english course. *Computers & Education*, 150:103839, 2020.
- 10 Andrzej Marczewski. User types. Retrieved from <https://www.gamified.uk/user-types/>, 2015.
- 11 John D. McCarthy, Kevin McCarthy, and Barry Smyth. User Type Taxonomies for Personalised Interactive Systems. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. ACM, 2022. doi:10.1145/3450439.3480504.
- 12 Nidhi Niyati, Pranav Awati, Neelam V Alai, and Ritu Jain. Escape rooms as an interactive learning tool: A systematic review. *International Journal of Emerging Technologies in Learning (iJET)*, 16(5):223–240, 2021.
- 13 Ricardo Alexandre Queiros, Mário Pinto, Alberto Simões, and Carlos Filipe Portela. A primer on gamification standardization. In Filipe Portela and Ricardo Queirós, editors, *Next-Generation Applications and Implementations of Gamification Systems*, pages 1–13. IGI Global, 2022. doi:10.4018/978-1-7998-8089-9.ch001.
- 14 Brittany Shapiro and Joshua Levine. The gamification of learning: A meta-analysis. *Journal of Educational Psychology*, 111(5):757–772, 2019.
- 15 Joke Voogt, Ola Erstad, Chris Dede, and Punya Mishra. Challenges to learning and schooling in the digital networked world of the 21st century. *Journal of Computer Assisted Learning*, 32(3):200–213, 2016.

# A Systematic Review of Formative Assessment to Support Students Learning Computer Programming

Jagadeeswaran Thangaraj ✉ 

School of Computing, Dublin City University, Ireland

Monica Ward ✉ 

School of Computing, Dublin City University, Ireland

Fiona O’Riordan ✉ 

Teaching Enhancement Unit, Dublin City University, Ireland

---

## Abstract

Formative assessment aims to increase student understanding, instructor instruction, and learning by providing feedback on students’ progress. The goal of this systematic review is to discover trends on formative assessment techniques used to support computer programming learners by synthesizing literature published between 2013 and 2023. 17 articles that were peer-reviewed and published in journals were examined from the initial search of 197 studies. According to the findings, all the studies were conducted at the higher education level and only a small number at the secondary school level. Overall, most studies found that motivation, scaffolding, and engagement were the three main goals of feedback, with less research finding that metacognitive goals were the intended outcomes. The two techniques for facilitating formative feedback that were used most frequently were compiler or testing based error messages and customised error messages. The importance of formative feedback is highlighted in the reviewed articles, supporting the contention that assessments used in programming courses should place a heavy emphasis on motivating students to increase their level of proficiency. This study also suggests a formative assessment that employs an adaptive strategy to evaluate the ability level of the novice students and motivate them to learn programming to acquire the necessary knowledge.

**2012 ACM Subject Classification** Applied computing → Education; Social and professional topics → Computing education; Social and professional topics → Student assessment

**Keywords and phrases** Automatic assessment, Computer programming, Formative assessment, Higher education, Novice programmer, Systematic review

**Digital Object Identifier** 10.4230/OASICS.ICPEC.2023.7

## 1 Introduction

Any course in a higher education institution worldwide that is concerned with software development requires programming modules. By introducing syntax and semantics, these modules aim to impart fundamental knowledge of programming languages [58]. Novice programmers are those taking their first computer programming courses or those with no prior programming experience. Independent components of programming will increase the difficulties of novices [42, 32]. Novice programmers are unable to interpret program code and have a lack of understanding of programming principles [27]. Although the computer science courses are in high demand, introductory programming modules frequently have dropout and failure rates as high as 50% [36, 34]. These modules play an important role to make them comfortable in continuing their education in computing [46]. Their interest in programming will rise once pedagogical methods motivate their confidence, and dropout rates will reduce [36].



© Jagadeeswaran Thangaraj, Monica Ward, and Fiona O’Riordan;  
licensed under Creative Commons License CC-BY 4.0

4th International Computer Programming Education Conference (ICPEC 2023).

Editors: Ricardo Alexandre Peixoto de Queirós and Mário Paulo Teixeira Pinto; Article No. 7; pp. 7:1–7:13

OpenAccess Series in Informatics



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

An essential component of education that promotes learning is assessment [48]. Additionally crucial for assisting and motivating their programming abilities are assessment and feedback [57]. Formative assessment aims to increase student understanding, instructor instruction, and learning by providing feedback on students' progress [12]. Formative programming assessment system evaluates student program submissions and provides timely feedback [42]. The impact of feedback on learning and assessment is significant [55]. Formative feedback is information given to a student with the goal of changing their way of thinking or acting to enhance learning [50]. Formative assessment is one of the approaches for effective programming learning [53]. To ensure that students receive the correct results, the assessment and feedback systems must examine the programs' small aspects and point out any areas where mistakes were made by the students. Beyond assessment, it increases novice programmers' self-confidence and metacognitive awareness [41, 49, 29]. Additionally, checking a participant's knowledge depending on their prior attempts during the assessment process is referred to as adaptive assessment [39]. It varies from typical assessment in that each participant receives a separate set of questions instead of everyone receiving the same set of questions [56]. This enables everyone to assess the knowledge at their own pace and helps in assessing each element of the topic. While some systematic reviews have been undertaken on automated assessments in computer programming, this study is interested in knowing whether formative assessment is used to scaffold or encourage novice programmers as feedback is essential to learning programming. It also examines any assessment system that makes use of an adaptive strategy.

## **2 Related Works**

Recent systematic literature reviews on assessment systems for programming courses tend to concentrate on how useful they are for automatic assessment techniques [24, 40, 20] or on the kind of feedback generated by assessment tools for evaluating programming languages or programming paradigms [25, 9, 28]. Further, other research reviews examined the assessment process, finding that it was primarily concentrated on advanced courses [35] or other computing subjects [16]. Studies that use automatic assessment programming as a pedagogical strategy typically focus solely on teaching and learning and do not examine findings on inspiring or motivating novices. In summary, there are no studies that concentrate on the formative assessment of introductory programming courses and no research reports that motivate novice programmers. Since motivation, scaffolding, and metacognitive support are important for learning programming modules, this research therefore examines formative assessment methods in these areas.

### **Research Questions**

This review of formative assessments for introductory programming is guided by the following research questions:

- RQ1: What is the purpose of the formative feedback techniques for programming languages learning?
- RQ2: What is the nature of the formative feedback techniques for programming languages learning?
- RQ3: Does the assessment method prioritize helping novices and influencing their programming learning?

■ **Table 1** Key Search Terms.

Key Concepts	Search Terms
Computer Programming	“novice computer programmers” or “computer programming” or “programming skills” or “first year computer programming” or “programming concepts” or “introductory programming” or “automated system” or “Automated assessment”
Formative Assessment	“formative feedback” or “error correction” or “learner confidence” or “learner efficacy” or “adaptive assessment” or “computer programming assessment system”

### 3 Research Methodology

More study, according to [14], is necessary to fully understand the inquiry process (formative assessment) that takes place in a programming learning environment. In order to better comprehend the current state of research, this study will examine recent articles that were released between 2013 and 2023. This study used this time frame of the last 10 years because the review topic is only applicable to contemporary studies [37]. This research was guided by the following PICO question: “The impact of formative feedback to motivate novices in learning introductory programming”. The What Works Clearinghouse Procedures and Standards Handbook, Version 4.0, from the U.S. Department of Education’s Institute of Education Sciences served as the process [33]. The five stages were as follows: (a) creating the review methodology; (b) locating pertinent literature; (c) screening studies; (d) reviewing articles; and (e) reporting findings. This study follows these stages to systematically review the literature.

#### 3.1 Data Sources and Search Strategies

The terms “Formative assessment” and “Computer programming” were broadly entered into databases using the Title, Keyword, and Abstract search functions to look for published publications between the years 2013 and 2023. Table 1 shows the alternative terms for the key terms in the search. Academic Search Complete, ERIC Library, ACM, IEEE and Science Direct were the databases combined. This research used search engines like Google Scholar, ResearchGate, and Academia to find manuscripts [22]. Additionally, core conferences in computer education such as SIGCSE, CompEd, ITiCSE, UKICER and ICPEC are taken into account. 197 articles were found in the initial search results since 1998. Based on the inclusion and exclusion criteria, these papers were evaluated at the title, abstract, and full text levels. 17 articles were produced as a result out of 22, and they were coded for the systematic review. As they concentrated on basic computer science, summative assessment, or gaming techniques rather than clearly focusing on introductory programming, formative assessment, or evaluation approaches, 5 articles were eliminated [51, 54, 43, 21, 15].

#### 3.2 Inclusion and Exclusion Criteria

Each study must meet these screening requirements to be considered for this systematic review: Computer programming and formative assessment are the article’s primary foci, and its publication dates range from 2013 to 2023. Its publication type is original research from peer-reviewed journals, and its research methodology includes both quantitative and qualitative approaches with a clear methods section and results presentation. Its language

## 7:4 A Systematic Review of Formative Assessment

■ **Table 2** Inclusion and Exclusion Criteria.

Criterion	Inclusion	Exclusion
Timeframe	2013–2023	Prior to 2013 the focus in literature prior to this time was primarily on plagiarism.
Language	English	Non-English
Access	Full-text availability only	Only titles or abstracts available
Sample	Novice programmers	Programming languages, Web design, model development, advanced programming, visual or scratch programming
Type of publication	Peer-reviewed, original research, conference papers	Content that was not peer-reviewed or original
Focus of literature	Presenting findings that help design an enhanced formative assessment system for novice programmers (what works and doesn't work, and why).	Studies that present findings on summative assessment. Systems designed for gifted programmers.

is English, and its emphasis on formative assessments that are being used to support their programming learning. If a research study did not satisfy one or more of the inclusion requirements that shows in Table 2, it was excluded.

### 4 Data Coding

Content analysis was used to find categorical themes from narrative data used to inform research focus and feedback strategy. In order to draw logical conclusions, content analysis aims to organize and interpret the data collected [6, 26]. The data gathering method across all research was formative or automated assessment. All studies took place in either a higher education or secondary context. All of the research was carried out globally, primarily in Northern America and Europe.

#### 4.1 Formative Feedback Purpose: (RQ1)

Using the main subcategories of formative feedback found in the chosen publications, authors expand the following categories as they are interrelated in achieving learning goals [1].

##### 4.1.1 Scaffold

The term “scaffolding” describes how an assessment plan might lead students through the steps of a larger project, with the teacher acting as an experienced leader who offers advice along the way [44]. The design of an assessment can scaffold the steps of a larger project by asking students to complete the steps so they can receive formative feedback in between [19].

##### 4.1.2 Motivation

A student’s motivation is defined as their “willingness, need, desire, and necessity to engage in and be successful in the learning process” [52]. The two main categories of motivational factors are intrinsic and extrinsic [2]. Self-motivation is another name for intrinsic motivation, which is the strong desire to learn a subject. Extrinsic motivation occurs when actions are

■ **Table 3** Purpose of formative feedback.

Instructional strategies	#	Studies
Scaffolding	5	[17][5][45][8][29]
Motivation	6	[19][38][4][3][23][31]
Engagement	4	[18][7][11][13]
Metacognitive/ Self-efficacy	2	[29][49]

taken to satiate an outside demand or receive an outside-imposed reward. Instead of just enjoying the activity or engaging in it for its own sake, they may be performed for their instrumental benefit.

### 4.1.3 Metacognitive and Self-efficacy

Understanding how to learn, participating actively, and reflecting on that engagement is defined as metacognition [10]. An individual’s belief in their own ability to do well is known as self-efficacy [2]. For successful learning programming, metacognition and self-efficacy are crucial abilities [30].

### 4.1.4 Engagement

Engagement promotes learning and forecasts students’ success [1]. It is a multidimensional meta-construct with elements of behavior, emotion, and cognition. Formative assessment helps students become more engaged in their learning and that makes them more confident in the subject [47].

## 4.2 Nature of Formative Feedback: (RQ2)

Authors distinguish between the studies using feedback techniques tailored to engage students in learning programming as follows:

### 4.2.1 Assessment Approach

This defines the assessment approaches to generate the feedback such as, self-assessment, peer assessment, (semi-) automated assessment on programming assignment.

### 4.2.2 Feedback Types

It defines what type of feedback the tool or system provides such as, standard error messages, customised error messages, testing report or grades.

### 4.2.3 Feedback Mechanism

It defines how the feedback is generated to notify such as unit testing, test cases, verification and validation, guided instructions or directions of tests.

■ **Table 4** Nature of formative feedback.

#	Studies	Assessment approaches	Feedback types	Feedback Mechanisms
4	[5, 13, 17, 4]	Automatic	Test case reports w/grade	Automated testing
1	[18]	Automatic	Unit test results	Automated testing
2	[3, 11]	Automatic	Verification report	(Static) Verifier generated
2	[31, 53]	Peer	Customised error messages	Peer (Lecturer/fellow) feedback
1	[29]	Self	Customised error messages	Rubric-based
3	[19, 38, 8]	Automatic	Standard error messages	Compiler
1	[49]	Peer	Customised error messages	Guided inquiry
2	[23, 7]	Semi-Automatic	Customised error messages	Feedback about code quality
1	[45]	Semi-Automatic	Customised error messages	Manual feedback

## 5 Results

### 5.1 Purpose of Formative Feedback (RQ1)

In order to determine the aim and different kinds of formative feedback strategies for learning programming languages, authors examined the relevant research studies in order to get the answers to the research questions. Overall, the majority of studies found that motivation, scaffolding, and engagement were the purposes of feedback, with metacognitive purposes being recognized in fewer studies (see Table 3). In general, the goal of all these studies is to inspire students who are learning programming at various levels.

### 5.2 Feedback Strategies (RQ2)

Automated assessment was more commonly employed to generate feedback rather than using other approaches. Customised error messages were provided for facilitating formative feedback that were most frequently used in addition to automated testing results and compiler-based standard error messages. They were customised by peer, manual feedback and rubric-based feedback. Guided inquiry, feedback on the quality of the code, and chatbot interaction are among the assessment mechanisms (see Table 4). In general, most systems offer customized error messages to help students comprehend the errors they committed.

### 5.3 Novices' Support (RQ3)

All of these studies generally aim to motivate students learning programming at different stages. This analysis found that, with the exception of one, no studies have focused especially on novices [45]. The assessment system that aids novices, however, makes use of Teaching Assistants' (TA's) feedback rather than automatic feedback [45]. As a result, no automatic formative feedback is primarily emphasizing novices to encourage their learning programming.

## 6 Discussion

Formative assessment systems offer automatic formative feedback to students who submit their solutions or programs. These systems use various technologies to provide this feedback. Students can receive automatic formative feedback based on automated unit and system tests in this system if the code is valid [18]; otherwise, they receive error messages so they can change it. This system tested how well formative feedback connects with students of diverse backgrounds and showed how it facilitated their learning of programming. Another system [17] employed laboratory exercises were accompanied by automated test cases created by the lecturer using solely free software testing tools that match industry standards in order to provide students with formative feedback as they were working. The immediate feedback and continued efforts to close the performance gap between actual and expected performance were lauded by the students, and the efficacy was determined to be successful [17].

The autoCOREctor, a tool for automated student-centered assessment, was created to be easily connected with learning management systems (LMSs)[5]. It encourages the development of a problem-statement scaffold for programming assignments and a straightforward test set with test cases to assist teachers in using it in diverse situations. Regarding the codes they entered, which they must attempt several times to pass, students receive grades and feedback. Because of this, the autoCOREctor’s feedback was helpful, easy to understand, helped students improve their assignments, and increased their motivation. The work’s drawback was that it was unclear whether it would be helpful for novice programmers or the introductory programming module.

A study looked at Algo+ and EPFL, two automated evaluation methods for online introductory programming courses [8]. Based on the discrepancy between the supplied program and the referent solution that is the closest match, Algo+ delivers comments. This distinction clarifies to the learner the processes to follow in order to arrive at the correct response. The feedback given by the EPFL grader is based on test cases and a check style process. The generated feedback educates students about the test case successes and failures by displaying the program’s result and the anticipated output. The student’s understanding of why the programs don’t provide the right responses even though they are syntactically correct is improved by the feedback on the most well-liked incorrect programs.

Incorporating online coding tasks into formative assessment is examined in an article to determine its practicality and efficacy [4]. Positive results from the experimental investigation back up the use of online coding environments in introductory programming and algorithm courses. It argues that formative rather than summative assessments enhance the learning process for students. As tutoring systems, chatbots have been used in a variety of settings. A system aims to use chatbots to teach basic CS concepts while increasing work completion and engagement among students, especially female students [7]. Because they produce formative feedback immediately at the task level and use the input to guide students toward learning programming, chatbots are useful tools for formative feedback.

Another formative assessments technique is that formative evaluation is combined with automatic source code verification and validation feedback [3]. With the use of this system, formative assessments will be able to provide feedback on the verification outcomes. When errors are discovered during the automatic verification phase, students will be given a report, enabling them to both fix the errors and gain a deeper understanding of the code. Similar system evaluated a library for automated assessments created especially for static analysis [11]. The lecturer can personalize exercises, reuse verification, and modify the lesson for each student using the library. It showed how flexible feedback on verification helps students identify inefficient and incorrect code fragments and encourages them to adopt good programming techniques.

Guided inquiry learning (GIL), an illustration of an inductive collaborative learning strategy. Students are expected to complete the learning objectives and provide their peers and the instructor comments on the problem [31]. It showed how receiving peer feedback improved their programming skills. Another system that used formative evaluation based on peer code review caused students' programming abilities to consistently improve [53]. Peer code review and inspection is an effective strategy to ensure the high quality of a software by methodically examining the source code. With the use of peer feedback, the students were able to identify and correct their errors. Similarly, a study looked at how students in introductory courses who are not majoring in computer science respond to evaluation situations [45]. The automatic evaluation system was utilized with TA's support. Because of this, the manual's (TA's) input was useful but not always practical to access.

A framework for understanding the what, why, and how of formative assessment of introductory programming in K–12 computer science was developed in order to answer the overall need for understanding formative assessment [19]. Thus, CS research on assessment design and programming learning, particularly student misconceptions, has an impact on the formative assessment questions' design [19]. Another study focused on how repeat questions can give students rapid feedback by using an internet platform called HERA [38]. It argues that the formative feedback was important and helpful for computational thinking [38]. According to another study [23], suggestions about unit length, unit complexity, and code duplication were the most beneficial to students. While this feedback does not help with the assignment, it enhances understanding [23].

A study [29] looked at the role of self-assessment in computer programming. Interest in learning is developing as a result of self-efficacy. The results of this study indicate that formative self-assessment may improve students' performance in an introductory programming course. Exercises in self-assessment with a rubric might be beneficial for first-year students. It was found that students who got comprehensive feedback on their learning were more motivated than those who merely got a rubric-based evaluation. According to this study, however, it was discovered that the self- assessment intervention had a practically significant impact on students' performance on programming projects. Another study [49] looked at the effects of open-ended assessment on students learning introductory programming in terms of performance and self-efficacy. Students who routinely completed the open-ended versions had higher average self-efficacy scores and assignment marks, though not by a statistically significant amount.

Because they can accommodate an endless number of students and submissions, Automated Testing and Feedback (ATF) systems were examined in this study to meet the demand [13]. The learning process can be completed by the student by submitting a novel answer and promptly receiving feedback. Feedback can address syntax errors, output accuracy, code performance, and if the code adheres to instructions exactly. The engagement and learning behaviors of learners in massive open online course (MOOCs) are examined in this study. This study found that code feedback is one of the most crucial aspects of MOOCs for programming and that there might be a positive trend toward ATF users getting better grades.

## **7** Conclusion

This systematic review revealed that most systems used customised feedback for formative assessment which adds more scaffolding to support learners' progression to the next level [13, 19]. In these systems, if the code is correct, students can receive automatic formative

feedback based on automated unit and system tests; if not, they receive error messages to fix it out [18, 5]. Students praised the quick feedback and the ongoing drive to decrease the performance gap between actual and intended performance, and the efficacy was rated successful [17]. The study’s positive findings support the use of formative assessments for introductory programming courses, and it makes the case that formative rather than summative assessments improve student experience [4]. Verifier generated feedback enabled the students to recognize and fix the errors using verification techniques [11, 3]. Students’ programming skills steadily increased because of formative assessment based on peer code review [53, 31]. When an automated assessment system was incorporated with manual feedback, the result was more beneficial, but it was not always realistically accessible [45]. Self-efficacy has attracted growing interest in learning. The findings of this study suggest that formative self-assessment may enhance students’ performance in a course on basic programming [49]. It was discovered that students who received granular feedback during their learning were more motivated than those who only received evaluation using a rubric and open-ended questions [29]. The positive is all these studies found the formative feedback (customised or standard error messages) were helpful to motivate, scaffold, engage or self-assess the learners in learning programming [23, 7, 29, 49, 45].

## 7.1 Limitations

However, only a few studies focused on novice learners and introductory programming [29, 45, 8]. There is no clear evidence that these formative assessments were helpful in motivating specifically the novice learners in programming assignments. By showing the result of the program that was submitted and the anticipated output, the generated feedback instructs students about the test case’s success or failure [8]. The work’s limitation was that it did not say whether it was useful for novice programmers who were just starting out or the introductory programming module [5]. Another limitation is that all these systems assess the same questions to all students. It does not support students with different abilities. Adaptive techniques are used in formative assessment to achieve its goals [56]. When a student provides an erroneous response to a question, the system can progressively lead the student through a discovery process that results in the proper solution, breaking down complex concepts one step at a time [35]. When students fail, it aids in providing assistance and encourages personalised learning [59].

## 7.2 Implications

This study provides an overall analysis of the formative assessment that underpins the programming module in various educational settings. For several factors, including scaffold, motivation, self-confidence, and engagement, enormous amounts of evidence was discovered. The utilization of feedback techniques and student participation in formative assessment, we discovered, had an impact. The findings of this review also suggest that several variables may have an impact on the various formative assessment strategies. To better assist novice programmers in learning programming, formative assessment needs to improve how it presents error messages. We could not find any research that addressed formative assessment or the use of feedback, despite the fact that these elements are probably crucial for inspiring novice learners. There is also less support for several criteria including adaptive strategy, purely because fewer research have looked into them. As a conclusion, this study recommends on how to use an adaptive strategy in the process of formative assessment, in order to especially motivate novices and boost their knowledge and confidence. Therefore, this study’s next work

will design a formative assessment system that uses the adaptive strategy with enhanced error messages to evaluate the ability level of the novice students and motivate them to learn programming to acquire the necessary knowledge.

---

## References

- 1 Elizabeth Acosta-Gonzaga and Aldo Ramirez-Arellano. Scaffolding Matters? Investigating Its Role in Motivation, Engagement and Learning Achievements in Higher Education. *Sustainability*, 14(20), 2022. doi:10.3390/su142013419.
- 2 Francisca A. Adamopoulos. *Learning Programming, Student Motivation*, pages 1–10. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-319-60013-0\_182-1.
- 3 Felipe I. Anfurrutia, Ainhoa Álvarez, Mikel Larrañaga, and Juan-Miguel López-Gil. Integrating Formative Feedback in Introductory Programming Modules. *IEEE Revista Iberoamericana de Tecnologías del Aprendizaje*, 13(1):3–10, 2018. doi:10.1109/RITA.2018.2801898.
- 4 Dhakshina Moorthy Anitha and Dhakshina Moorthy Kavitha. Online coding event as a formative assessment tool in introductory programming and algorithmic courses —A exploration study. *Computer Applications in Engineering Education*, 28:1580–1590, 2020.
- 5 Enrique Barra, Sonsoles López-Pernas, Álvaro Alonso, Juan Fernando Sánchez-Rada, Aldo Gordillo, and Juan Quemada. Automated Assessment in Programming Courses: A Case Study during the COVID-19 Era. *Sustainability*, 12(18), 2020. doi:10.3390/su12187451.
- 6 Mariette Bengtsson. How to plan and perform a qualitative study using content analysis. *NursingPlus Open*, 2:8–14, 2016. doi:10.1016/j.npls.2016.01.001.
- 7 Luciana Benotti, Mara Cecilia Martnez, and Fernando Schapachnik. A Tool for Introducing Computer Science with Automatic Formative Assessment. *IEEE Transactions on Learning Technologies*, 11(2):179–192, 2018. doi:10.1109/TLT.2017.2682084.
- 8 Anis Bey, Patrick Jermann, and Pierre Dillenbourg. A Comparison between Two Automatic Assessment Approaches for Programming: An Empirical Study on MOOCs. *Journal of Educational Technology & Society*, 21(2):259–272, 2018. URL: <http://www.jstor.org/stable/26388406>.
- 9 Sébastien Combéfis. Automated Code Assessment for Education: Review, Classification and Perspectives on Techniques and Tools. *Software*, 1(1):3–30, 2022. doi:10.3390/software1010002.
- 10 David T. Conley and Elizabeth M. French. Student Ownership of Learning as a Key Component of College Readiness. *American Behavioral Scientist*, 58(8):1018–1034, 2014. doi:10.1177/0002764213515232.
- 11 Pedro Delgado-Pérez and Inmaculada Medina-Bulo. Customizable and scalable automated assessment of C/C++ programming assignments. *Computer Applications in Engineering Education*, 28:1449–1466, 2020.
- 12 Suzanne W. Dietrich, Don Goelman, Jennifer Broatch, Sharon M. Crook, Becky Ball, Kimberly Kobjek, and Jennifer Ortiz. Using Formative Assessment for Improving Pedagogy: Reflections on Feedback Informing Database Visualizations. *ACM Inroads*, 11(4):27–34, November 2020. doi:10.1145/3430766.
- 13 Hagit Gabbay and Anat Cohen. Investigating the effect of Automated Feedback on learning behavior in MOOCs for programming. In *Proceedings of the 15th International Conference on Educational Data Mining*, pages 376–383. International Educational Data Mining Society, July 2022. doi:10.5281/zenodo.6853125.
- 14 Donn Randy Garrison. *E-learning in the 21st century: A community of inquiry framework for research and practice*. Third Edition, October 2016. doi:10.4324/9781315667263.
- 15 Ashok Goel and David Joyner. Formative Assessment and Implicit Feedback in Online Learning. In *Proceedings of Learning with MOOCs III, Philadelphia, PA*, 2016. URL: <https://www.davidjoyner.net/blog/formative-assessment-and-implicit-feedback-in-online-learning/>.

- 16 Rubén González. ICE: An Automated Tool for Teaching Advanced C Programming. *International Association for Development of the Information Society*, 2017.
- 17 Peadar F. Grant. Formative test-driven development for programming practicals. *AISHE-J: The All Ireland Journal of Teaching and Learning in Higher Education*, 9, 2017.
- 18 Beate Grawemeyer, John Halloran, Matthew England, and David Croft. Feedback and Engagement on an Introductory Programming Module. *CoRR*, abs/2201.01240, 2022. [arXiv: 2201.01240](https://arxiv.org/abs/2201.01240).
- 19 Shuchi Grover. Toward A Framework for Formative Assessment of Conceptual Learning in K-12 Computer Science Classrooms. In *Association for Computing Machinery, SIGCSE ’21*, pages 31–37, New York, NY, USA, 2021. doi:10.1145/3408877.3432460.
- 20 Sugandha Gupta and Anamika Gupta. E-Assessment Tools for Programming Languages: A Review. In *International Conference on Information Technology and Knowledge Management*, 2018.
- 21 Thomas Hainey, Gavin Baxter, Julie Black, Kenneth Yorke, Julius Bernikas, Natalia Chrzanowska, and Fraser McAulay. Serious games as innovative formative assessment tools for programming in higher education. In *ECGBL, 16th European Conference on Games Based Learning, 6 - 7 October 2022, Lisbon, Portugal*, June 2022. URL: <https://www.academic-conferences.org/conferences/ecgbl/>.
- 22 Jared Howland, Thomas Wright, Rebecca Boughan, and Brian Roberts. How Scholarly Is Google Scholar? A Comparison to Library Databases. *College & Research Libraries*, 70:227–234, May 2009. doi:10.5860/cr1.70.3.227.
- 23 Julian Jansen, Ana Oprescu, and Magiel Bruntink. The impact of automated code quality feedback in programming education. In *Post-proceedings of the Tenth Seminar on Advanced Techniques and Tools for Software Evolution (SATToSE)*, volume 210, 2017.
- 24 Maria Kallia. Assessment in Computer Science courses: A Literature Review. In *King’s College London*, 2017.
- 25 Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. *ACM Transactions on Computing Education*, 19:1–43, September 2018. doi:10.1145/3231711.
- 26 Barbara Kitchenham and Stuart Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. *Keele University and University of Durham technical report*, 2, January 2007.
- 27 Melisa Koorsse, Charmain Cilliers, and André P. Calitz. Programming assistance tools to support the learning of IT programming in South African secondary schools. *Comput. Educ.*, 82:162–178, 2015.
- 28 Nguyen-Thinh Le. A Classification of Adaptive Feedback in Educational Systems for Programming. *Systems*, 4(2), 2016. doi:10.3390/systems4020022.
- 29 Alex Lishinski and Aman Yadav. Self-evaluation Interventions: Impact on Self-efficacy and Performance in Introductory Programming. *ACM Transactions on Computing Education (TOCE)*, 21:1–28, 2021.
- 30 Dastyni Loksa, Lauren Margulieux, Brett A. Becker, Michelle Craig, Paul Denny, Raymond Pettit, and James Prather. Metacognition and Self-Regulation in Programming Education: Theories and Exemplars of Use. *ACM Trans. Comput. Educ.*, 22(4), September 2022. doi:10.1145/3487050.
- 31 Jose Manappattukunnel Lukose and Kuttickattu John Mammen. Enhancing Academic Achievement in an Introductory Computer Programming Course through the Implementation of Guided Inquiry-Based Learning and Teaching. In *Asia-Pacific Forum on Science Learning and Teaching*, volume 19,2, 2018.
- 32 Andrew Luxton-Reilly, Brett A. Becker, Yingjun Cao, Roger McDermott, Claudio Mirolo, Andreas Mühlhling, Andrew Petersen, Kate Sanders, Simon, and Jacqueline Whalley. Developing Assessments to Determine Mastery of Programming Fundamentals. In *Association for Computing Machinery, ITiCSE-WGR ’17*, pages 47–69, New York, NY, USA, 2018. doi:10.1145/3174781.3174784.

- 33 Daniel M. Maggin, Erin Barton, Brian Reichow, Kathleen Lynne Lane, and Karrie A. Shogren. Commentary on the What Works Clearinghouse Standards and Procedures Handbook (v. 4.1) for the Review of Single-Case Research. *Remedial and Special Education*, 43(6):421–433, 2022. doi:10.1177/07419325211051317.
- 34 Sohail Iqbal Malik and Jo Coldwell-Neilson. A model for teaching an introductory programming course using ADRI. *Education and Information Technologies*, 22:1089–1120, 2017.
- 35 Marina Marchisio, Tiziana Margaria, and Matteo Sacchet. Automatic Formative Assessment in Computer Science: Guidance to Model-Driven Design. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 201–206, 2020.
- 36 Lauren E. Margulieux, Briana B. Morrison, and Adrienne Decker. Reducing withdrawal and failure rates in introductory programming with subgoal labeled worked examples. *International Journal of STEM Education*, 7:1–16, 2020.
- 37 Timothy Meline. Selecting Studies for Systemic Review: Inclusion and Exclusion Criteria. *Contemporary Issues in Communication Science and Disorders*, 33(Spring):21–27, 2006. doi:10.1044/cicsd\_33\_S\_21.
- 38 Laura Orozco-Garcia, Carolina Gonzalez, Juan Montano, Cristian Mondragon, and Hendrys Tobar-Munoz. A Formative Assessment Tool to Support Computational Thinking in the Classroom. In *2019 International Conference on Virtual Reality and Visualization (ICVRV)*, pages 185–188, 2019. doi:10.1109/ICVRV47840.2019.00043.
- 39 Elena C. Papanastasiou. *Adaptive Assessment*, pages 18–19. Springer Verlag, 2015.
- 40 Raymond Pettit, J.D. Homer, K.M. Holcomb, N. Simone, and Susan Mengel. Are automated assessment tools helpful in programming courses? *ASEE Annual Conference and Exposition, Conference Proceedings*, 122, January 2015.
- 41 James E. Prather. Beyond Automated Assessment: Building Metacognitive Awareness in Novice Programmers in CS1. In *Nova Southeastern University*, 2018.
- 42 Yizhou Qian and James Lehman. Students’ Misconceptions and Other Difficulties in Introductory Programming: A Literature Review. *ACM Trans. Comput. Educ.*, 18(1), October 2017. doi:10.1145/3077618.
- 43 Yizhou Qian and James Lehman. Using an automated assessment tool to explore difficulties of middle school students in introductory programming. *Journal of Research on Technology in Education*, 54:1–17, January 2021. doi:10.1080/15391523.2020.1865220.
- 44 Brian J. Reiser and Iris Tabak. *Scaffolding*, pages 44–62. Cambridge University Press, United Kingdom, January 2014. doi:10.1017/CB09781139519526.005.
- 45 Emma Riese and Stefan Stenbom. Experiences of Assessment in Introductory Programming From the Perspective of NonComputer Science Majors. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE Press, 2020. doi:10.1109/FIE44824.2020.9274060.
- 46 Siti Nurulain Mohd Rum and Maizatul Akmar Binti Ismail. Metacognitive Support Accelerates Computer Assisted Learning for Novice Programmers. *J. Educ. Technol. Soc.*, 20:170–181, 2017.
- 47 G. W. Scott. Active engagement with assessment and feedback can improve group-work outcomes and boost student confidence. *Higher Education Pedagogies*, 2(1):1–13, 2017. doi:10.1080/23752696.2017.1307692.
- 48 Nicole Shanley, Florence Martin, Nicole Collins, Manuel Perez-Quinones, Lynn Ahlgrim-Delzell, David Pugalee, and Ellen Hart. Teaching Programming Online: Design, Facilitation and Assessment Strategies and Recommendations for High School Teachers. *TechTrends*, 66, April 2022. doi:10.1007/s11528-022-00724-x.
- 49 Sadia Sharmin, Daniel Zingaro, Lisa Zhang, and Clare Brett. Impact of Open-Ended Assignments on Student Self-Efficacy in CS1. In *CompEd ’19: Proceedings of the ACM Conference on Global Computing Education*, pages 215–221, April 2019. doi:10.1145/3300115.3309532.
- 50 Valerie J. Shute. Focus on Formative Feedback. *Review of Educational Research*, 78(1):153–189, 2008. doi:10.3102/0034654307313795.

- 51 Judith Stanja, Wolfgang Gritz, Johannes Krugel, Anett Hoppe, and Sarah Dannemann. Formative assessment strategies for students’ conceptions—The potential of learning analytics. *British Journal of Educational Technology*, 54(1):58–75, 2023. doi:10.1111/bjet.13288.
- 52 Ricarda Steinmayr, Anne F. Weidinger, Malte Schwinger, and Birgit Spinath. The Importance of Students’ Motivation for Their Academic Achievement – Replicating and Extending Previous Findings. *Frontiers in Psychology*, 10, 2019. doi:10.3389/fpsyg.2019.01730.
- 53 Qing Sun, Ji Wu, Wenge Rong, and Wenbo Liu. Formative assessment of programming language learning based on peer code review: Implementation and experience report. *Tsinghua Science and Technology*, 24:423–434, August 2019. doi:10.26599/TST.2018.9010109.
- 54 Elise Trumbull and Andrea A. Lash. Understanding Formative Assessment Insights from Learning Theory and Measurement Theory. In *WestEd*, 2013.
- 55 Fabienne M. van der Kleij, Theodorus Johannes Hendrikus Maria Eggen, Caroline F. Timmers, and Bernard P. Veldkamp. Effects of feedback in a computer-based assessment for learning. *Comput. Educ.*, 58:263–272, 2012.
- 56 Jill-Jênn Vie, Fabrice Popineau, Éric Bruillard, and Yolaine Bourda. *A Review of Recent Advances in Adaptive Assessment*, volume 94, pages 113–142. Studies in Systems, Decision and Control, February 2017. doi:10.1007/978-3-319-52977-6\_4.
- 57 Xiao-Ming Wang, Gwo-Jen Hwang, Zi-Yun Liang, and Hsiu-Ying Wang. Enhancing students’ computer programming performances, critical thinking awareness and attitudes towards programming: An online peer assessment attempt. *Educational Technology and Society*, 20:58–68, January 2017.
- 58 Stelios Xinogalos, Tomáš Pitner, Miloš Savić, and Mirjana Ivanović. *First Programming Language in Introductory Programming Courses, Role of*, pages 1–11. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-319-60013-0\_217-1.
- 59 Albert Yang, Brendan Flanagan, and Hiroaki Ogata. Adaptive formative assessment system based on computerized adaptive testing and the learning memory cycle for personalized learning. *Computers and Education: Artificial Intelligence*, 3:100104, October 2022. doi:10.1016/j.caeai.2022.100104.



# A New Approach to Perform Individual Assessments at Higher Education Using Gamification Systems

Filipe Portela  

Algoritmi Centre, University of Minho, Guimarães, Portugal

---

## Abstract

Assessment is a crucial element of the educational process, but traditional pen-and-paper tests have limitations in promoting active learning and engagement. To address this challenge, the use of online gamification platforms has increased. In this context, this study explores the effectiveness of Kahoot! for assessment exercises (AE) in higher education. These experiments occurred over three years, included five courses with computer science subjects and had 507 participants. Overall, 97.04% of students achieved a grade higher than ten, and only four failed. The results show that Kahoot! can promote engagement, motivation, and learning outcomes, and its use is well-received by students – 78.70% of students enjoyed this approach, and only 8.68% of participants disliked it. The study’s findings provide valuable insights into using Kahoot! as Student Response System for testing in higher education, with implications for developing new and innovative approaches to assessment and evaluation.

**2012 ACM Subject Classification** Information systems → Information retrieval

**Keywords and phrases** TechTeach, Information Systems, Gamification, Higher Education Assessment Tools, Kahoot!

**Digital Object Identifier** 10.4230/OASICS.ICPEC.2023.8

**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

Assessment is a critical component of the educational process, and the type of tests used is essential to its effectiveness. Traditional pen-and-paper tests are frequently utilised in higher education but have limitations. Engaging students and promoting active learning is a significant challenge. To address this issue, online platforms such as Kahoot! have been developed to provide an interactive and engaging environment for assessment. Online platforms in higher education emphasise the importance of incorporating active learning techniques into teaching to promote deeper understanding and engagement [17]. Active learning has been shown to have numerous benefits, including increased motivation, engagement, and better academic performance [12]. Online platforms like Kahoot! have the potential to facilitate active learning by providing an interactive and engaging environment for assessment.

Although alternatives arise, professors still focus on traditional learning with tests/exams and distrust the efficiency of new approaches/solutions. To solve it, the TechTeach paradigm [17] was created to turn classes more attractive and engaging using emerging techniques and technologies. In this context, some new experiences were performed to overcome the stigma created by online assessments. Using interactive tools (e.g. Mentimeter, VoxVote, Kahoot!, among others) in higher education Assessment Exercises (AE) can result in better engagement, higher motivation, and improved learning outcomes than traditional pen-and-paper tests. AE can include different types, such as tests, mini-tests, or exams.



© Filipe Portela;

licensed under Creative Commons License CC-BY 4.0

4th International Computer Programming Education Conference (ICPEC 2023).

Editors: Ricardo Alexandre Peixoto de Queirós and Mário Paulo Teixeira Pinto; Article No. 8; pp. 8:1–8:12

OpenAccess Series in Informatics



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

## 8:2 A New Approach to Perform Individual Assessments

This study explores a new assessment strategy proposed by professors and evaluates the effectiveness of an online platform – Student Response Systems (SRS) – for higher education AE. The traditional focus of testing students' knowledge with complex questions has shifted to promoting active learning by assessing whether students have assimilated the basics. The goal is to develop students' ability to understand the foundational concepts and enable them to comprehend problems and identify possible solutions. Professors summarise the matter during the test, explaining the answers and highlighting the most critical aspects. The assessment paradigm must change. With the abundance of platforms available, such as Google, Bing, or ChatGPT, professors can focus on teaching students how to find the information they need rather than memorising it.

This new approach was introduced in 2020/2021, and since then, Kahoot! has been used to assess knowledge from over 500 individual AE. These experiments were performed in various courses with computer science subjects, including civil engineering (CIV), Textile Engineering (TEXT), Applied Maths and Statistics (ESTAP), Data Science (DS) and Engineering and Management of Information Systems (EGSI). This work aimed to assess the feasibility of using Kahoot! to evaluate students' knowledge and promote active learning. This article presents the results of several experiments and discusses the implications of the authors' findings for using online platforms in higher education assessment.

This paper is structured into seven sections. The first section, Introduction, presents the goal of the work and the relevant background information. The second section, Background, provides an explanation of some key concepts. The third section, Materials and Methods, outlines the methodologies and tools used in this case study. Forth section presents the approach and gamification rules. The fifth section, Case Study, presents the experiments and results. The sixth section, Discussion, analyzes the results in detail. Finally, the last section, Conclusion, summarizes the study's main findings and provides recommendations for future research.

### **2** Background

This section presents the main topics of the work and some similar works.

#### **2.1 Academic Assessments**

Academic assessments have been an essential aspect of education for centuries, providing a means of evaluating student learning and ensuring that educational goals are being met. Over time, assessment practices have evolved to include a range of methods, including standardized tests, essays, projects, and presentations, among others [4, 5, 19].

However, the traditional approach to assessment has been criticized for its emphasis on high-stakes testing, which can create a culture of fear and anxiety among students and limit their ability to learn and grow. In response, alternative assessment methods that emphasize formative assessment, feedback, and student engagement have been proposed and implemented in various educational settings [5, 9].

One of the emerging alternative assessment methods is using gamification and online platforms, such as Kahoot!, in higher education assessments. These platforms have been found to enhance student engagement, motivation, and learning outcomes and provide instructors with real-time feedback on student performance [10, 6].

As the field of education continues to evolve, assessments will remain an integral part of the learning process. The ongoing development and adoption of innovative assessment practices, including gamification and online platforms (e.g. Student Response Systems), will undoubtedly continue to shape the future of education and improve student outcomes.

## 2.2 TechTeach

TechTeach is a new approach to enhancing student engagement in the classroom by using technology [17]. The authors describe it as combining various digital tools and approaches, such as Gamification, Bring Your Own Device (BYOD), B-learning or project and team-based learning to enhance student engagement. In the context of this work, Gamification and BYOD can be highlighted. Gamification is a different way to assess students, where professors can use the students' actions to give points. BYOD consists of using personal devices (e.g. computer or smartphone) in classes to interact with the subject or do practical exercises. TechTeach suggests that using technology in classrooms can effectively improve education quality and enhance students' learning experience [18].

## 2.3 Student Response Systems

Student Response Systems (SRS), also known as classroom response systems or clickers, are technology-based tools that enable instructors to engage and assess students in real-time during lectures or presentations. These systems allow students to respond to questions or prompts using handheld devices, such as clickers or mobile devices, and the responses are collected and displayed instantly for both the instructor and students to see [7].

## 2.4 Similar Works

Student Response Systems have become increasingly popular in higher education for their potential to facilitate active learning and engagement. One such platform is Kahoot!, a game-based learning platform that allows instructors to create quizzes and interactive activities that students can access via their devices.

Kahoot! has been used in various educational contexts, including primary, secondary, and higher education. In higher education, Kahoot! has been used as a tool for formative assessment, student engagement, and promoting active learning. Studies have shown that Kahoot! can improve learning outcomes and increase student motivation and engagement. [15, 13].

Several studies have evaluated the effectiveness of Kahoot! for testing in higher education. For example, Al-Busaidi et al. [1] evaluated the use of Kahoot! as a formative assessment tool in a medical school course and found that students were highly motivated and engaged with the platform. Similarly, study [16] found several works using Kahoot to engage and assess students. Other studies have compared Kahoot! to traditional pen-and-paper tests. For example, Chiang et al. [8] compared the effectiveness of Kahoot! and pen-and-paper tests in English as a foreign language class and found that Kahoot! resulted in higher scores and greater engagement.

Furthermore, a study by Hunsu and Adesope (2016) [14] conducted a meta-analysis of research on clicker use in higher education and found that clickers positively impacted student engagement, academic achievement, and overall learning outcomes.

While Kahoot! has shown promise as a tool for testing in higher education, it is not without limitations. For example, Kahoot! quizzes are typically short and may not be suitable for more complex topics. Additionally, Kahoot!'s gamification elements may not be appealing to all students.

Other systems similar to Kahoot! have also been used for testing in higher education. For example, Socrative and Quizlet have been used to assess student knowledge and promote active learning [2]. Like Kahoot!, these platforms provide an interactive and engaging environment for assessment.

## 8:4 A New Approach to Perform Individual Assessments

In summary, Kahoot! and SRS have shown potential as tools for testing in higher education. Studies have demonstrated their effectiveness in promoting engagement, motivation, and learning outcomes. However, further research is needed to explore their use in more complex topics and to determine their effectiveness over the long term.

### 3 Material and Methods

This article follows the case study methodology, a qualitative research method involving a detailed investigation of a particular phenomenon or event [20]. The case study methodology typically involves several phases: design, data collection, analysis, and interpretation [3]. The case study consisted in following phases and tasks:

- Design:
  - Creation of the exercise and AE based on the course's learning objectives and curriculum
  - Planning of questions and strategies used in the exercises
  - Definition of rules for calculating grades
  - Design of students' opinion questions to assess their perception of the Kahoot! platform for testing in higher education
- Implementation:
  - Administration of the designed exercise and AE to students using the Kahoot! platform
  - Collection of data on students' performance and grades according to the predefined rules
  - Use of students' opinion questions to assess their perception of the Kahoot! platform for testing
- Analysis:
  - Examination of collected data using various statistical methods, such as descriptive statistics and regression analysis
  - Determination of relationships between students' performance and the Kahoot! platform for testing
  - Analysis of students' opinion responses using thematic analysis to identify any recurring themes or patterns in their feedback [3]
- Interpretation:
  - Interpretation of results to draw conclusions and make recommendations for the use of Kahoot! in higher education testing
  - Comparison of findings with existing literature on the topic
  - Discussion of implications of the results in the broader context of higher education testing

This study follows the case study methodology to provide an in-depth analysis of the use of Kahoot! as an SRS for testing in higher education. By collecting and analyzing data on students' performance and perception of the platform, this study aims to contribute to the growing body of research on the use of online platforms for testing in higher education. The case study methodology is well-suited for exploratory research in real-world settings [11], making it an ideal approach for this study.

### 4 Approach

This section presents the new assessment approach designed (design phase of case study methodology) and then tested with the case study.

## 4.1 Gamification model

The Gamification model enhances a narrative that can be highlighted. It is a key aspect of this approach. The students must know the rules before starting each AE.

1. Students should have previous contact with the tool used (e.g. VoxBot, Kahoot! or other) and must test all question types before the AE start (e.g. short, true or false, multiple choice, among others).
2. All questions must have all details easily identified - question value (bonus), type and timing, left time, number of questions missing, and possible answers.
3. The questions of each AE should only address the fundamentals and essential subjects.
4. Questions must be designed according to the basic knowledge that students must have in their professional work. Subjects that a typical worker needs to use Google or similar should be avoided. However, it must be part of a question if they need to know something before using Google.
5. Each participant can answer the AE using BYOD: smartphone, computer or tablet.
6. The system should incorporate a cut-off value that allows for the exclusion of certain questions. For instance, if the percentage of correct answers to a particular question is low, it indicates that the question may be poorly formulated or that the professor may have been ineffective in explaining the corresponding content. In such cases, the final assessment should not include the question.
7. The Evaluation criteria should consider quickness and rightness. Each question is timed, so if a student answers with a response time inside of Q3 answers time, he receives 100%; otherwise, he receives a percentage according to the average time
8. If a question is relevant, it should have double points (200%)
9. Quick students with correct answers (higher than the average) can receive a bonus.
10. The question's time varies according to the typologies and complex level.
11. In case multiple answers are allowed, if students hit at least 50% of cases, they receive some points; otherwise, they have 0.

The professor can add or consider different ideas; however, they must be explained at the beginning of the subject.

## 4.2 Rules

The guidelines for the assessment exercise were developed as part of the TechTeach paradigm and considered a set of specific factors, including

- **Type of questions:** true and false; multiple choice; short answer; ordering
- **Time limited:** 10s, 30s, 60s, and 90s
- **Valuation:** normal and double
- **Knowledge:** essential and must have
- **Evaluation criteria:** quickness (quartiles and average time), rightness
- **Types of exercises:** simple and quick questions, code with variables or images.
- **Questions number:** 25-30 (a ratio from each matter taught) - Many questions allow crossing all matters addressed and show if students know all the basics.

This approach evaluates critical knowledge, and the AE class can be used for reviewing purposes. After the students answer each question, the professor should explain the question and respective answers to the class, ensuring that the key knowledge of the subject is not forgotten.

Strategies like questions with short time, bonuses and others can be used to avoid copy. Students are instigated to answer quickly with their knowledge; otherwise, time elapses, and they will not respond on time.

## 8:6 A New Approach to Perform Individual Assessments

A case study was designed and implemented to test this approach over the last three years.

### 5 Case Study

This case study started in 2020 and followed some TeachTeach guidelines [17], including gamification [18] and bringing your own device (BYOD), which was presented in Section 4. Since 2020, 507 students participated in this study and used Kahoot! to perform their exercises. The study was applied in several subjects(courses), including:

- **Introduction to Programming**

- Courses: Civil, Textile, Applied Statistics and Maths, Data Science
- Academic Year: 1st

- **Web Programming**

- Course: Information Systems
- Academic Year: 2nd and 3rd

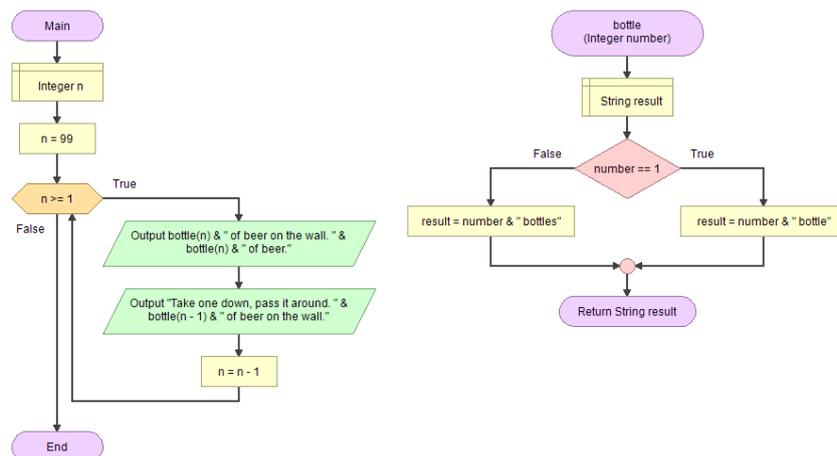
#### 5.1 Implementation

The following list presents the implementation phase and shows some questions that exemplify the rules explained in section 4.2. The list includes the question, its type, the defined answering time, and the available answer choices.

(A) **Question:** What is the “not equal to” symbol in Em Gaddys?

- **Question type:** short answer
- **Time-limited:** 30 sec
- **Solution:** <>

(B) **Question:** Which of the following loops (figure 1) is a “while” loop?



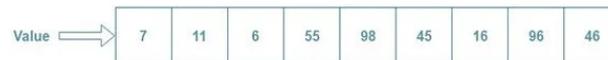
■ **Figure 1** Example of While Loop and IF.

- **Time-limited:** 30 sec
- **Question type:** Quiz
- **Answers:**
  - (a) A
  - (b) B
  - (c) None of the above

(C) **Question:** What does a compiler do?

- **Time-limited:** 30 sec
- **Question type:** Quiz
- **Answers:**
  - (a) Translates the source code instructions into Assembly language instructions
  - (b) Translates Assembly language instructions into corresponding binary code
  - (c) Prepares the object code to be loaded into memory and executed
  - (d) Examines, decodes, and executes each instruction of the source code line by line

(D) **Question:** What is the value of a[5] in figure 2?



■ **Figure 2** Example of array.

- **Question type:** Type answer
  - **Time-limited:** 30sec
  - **Solution:** 45
- (E) **Question:** How many columns does Bootstrap's grid system have?
- **Question type:** Type answer
  - **Time-limited:** 20 sec
  - **Possible Solutions:**
    - (a) 12
    - (b) twelve
    - (c) Twelve
- (F) **Question:** Order the following options in order to validate the code of figure 3

```

window.onload = pageLoad;

//chamado quando a página termina de ser carregada; sets up event handlers
function pageLoad() {
  var computeButton = document.getElementById("compute");
  computeButton.onclick = compute;
}

// Multiplies two numbers typed into input boxes on the page,
// and displays the result in a span on the page.
function compute() {
  var A = document.getElementById("num1"); //fetch the 2 numbers
  var input2 = document.getElementById("num2");
  var B = document.getElementById("answer");
  var C = input1.value * input2.value; // compute result
  answer.D = result;
}

```

■ **Figure 3** An example of code with missing details.

- **Question type:** Ordering
- **Time-limited:** 90sec

  - (a) input1
  - (b) answer
  - (c) result
  - (d) innerHTML

### 5.2 Analysis of the results

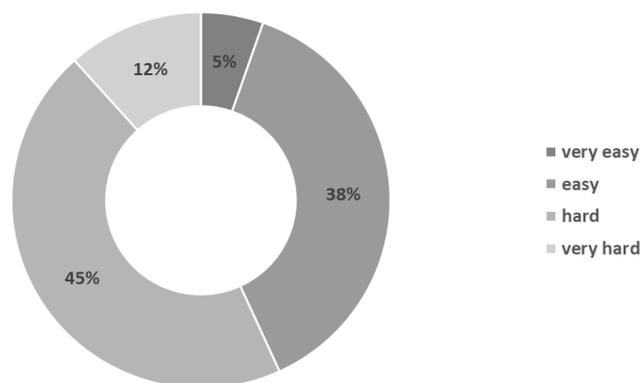
During the case study, which started in the academic year 2020/2021, seven distinct exams were conducted using Kahoot! the platform for testing in higher education. These exams were administered in multiple courses. 2020/21 was the team’s first year in this subject, and the team started by experimenting with this approach, so it was not possible to compare the results with traditional methods (pen & paper) yet. Regarding the results, three types of performance data were evaluated: quickness, rightness, and exercise relevance. The quickness of the students’ responses was measured using quartiles and average time, while rightness was calculated based on the number of correct answers. The exercise relevance was classified as a normal or double point.

In this case study, a cut-off was defined, i.e., the lower number of correct answers to each question ranged between five and twelve per cent. So, questions where the number of rightness was lower than the cut-off were removed.

To understand students’ opinions about this evaluation approach, the two last questions of each EA are

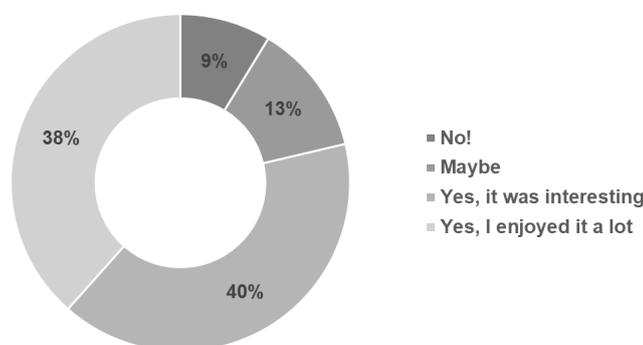
1. Q1 – How challenging was the examination?
2. Q2 – Did you approve this model?

Figure 4 depicts students’ opinions about the difficulty of the exam. As observed, more than 50% of the students considered it hard or very hard.



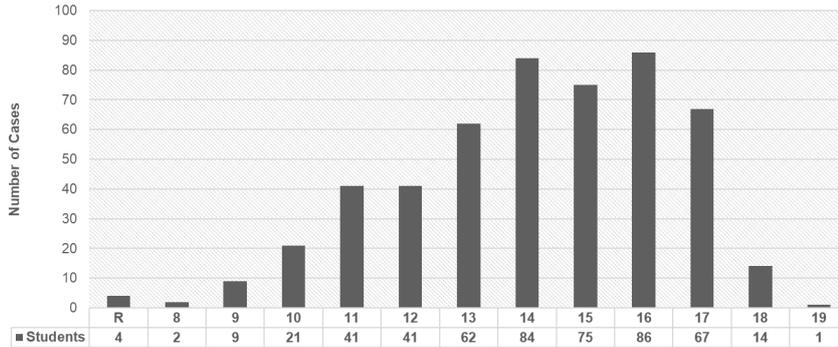
■ Figure 4 Q1 – How challenging was the examination?

Figure 5 illustrates the students’ approval of this exercise model. Of the total number of students surveyed, 78 approved of this model, while only 9 did not want this type of exercise.



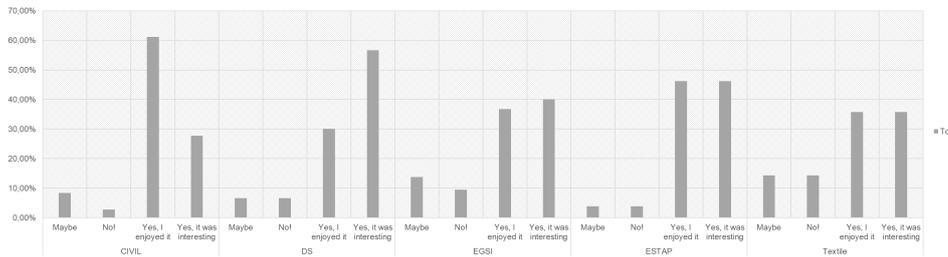
■ Figure 5 Q2 – Did you approve this model?

The following image (Figure 6) shows the distribution of student grades. As can be observed, this method resulted in only four students (0.78%) failing (grade R) and 2.96% receiving a negative rate of less than 10. Furthermore, the results indicate that this method efficiently avoids too high grades. Conscientious, we can understand that none of the students can know everything a professor teaches, so they hit answer all the questions. So, only 15 students (2.96%) achieved a grade between eighteen and twenty.



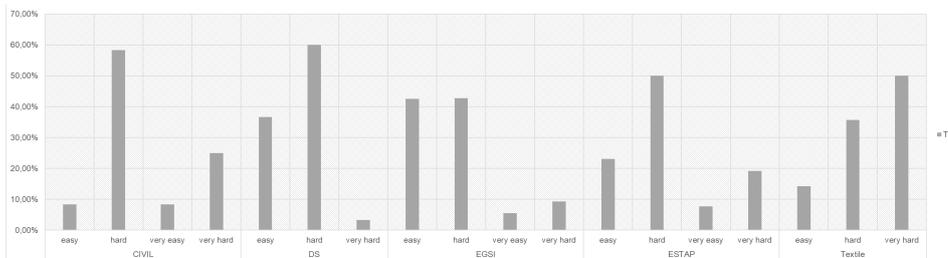
■ **Figure 6** Results distribution.

Another interesting analysis pertains to students' perceptions of the assessment exercises (figure 7), broken down by course. While generally, all courses expressed approval of this approach, certain courses such as CIVIL (>60%) demonstrated the highest level of approval (i.e., 'Yes, I enjoyed it')



■ **Figure 7** Approved opinion By Course.

The same analysis can be performed based on students' perceptions of the difficulty of the assessment exercise (Figure 8). Overall, the assessment exercise was perceived as difficult or very difficult by the students across all courses, with a minimum of 50% of students rating it as such.



■ **Figure 8** Difficulty perception by Course.

## 8:10 A New Approach to Perform Individual Assessments

Another potential avenue for further research is using pivot tables to analyze and compare students' opinions and grades across different courses.

Figure 9 presents the minimum (min), maximum (max), and average grades grouped by students' approval of the approach and their perception of the AE difficulty. This figure crosses user expectations, perceptions and the achieved results. For instance, in the Civil course, students who enjoyed the AE and found it challenging achieved grades ranging from 8.30 to 15.18. Students who enjoyed the mechanism but considered it difficult could achieve good grades.

In another analysis, EGSI students who may approve this mechanism and consider the exam easy had a minimum grade of 9,18 and a maximum grade of 17,03 with an average of 14.07. Globally, those who considered the AE easy achieved better grades than others.

Curiously, some students who enjoyed the mechanism did not achieve positive grades, which means the agent is correct, but they must study more to achieve better results.

This analysis can provide valuable insights into the relationship between students' opinions, perceived difficulty, and their academic performance in different courses.

Row Labels	Min of RESULT	Max of RESULT	Average of RESULT
<b>CIVIL</b>	<b>8,30</b>	<b>15,18</b>	<b>11,84</b>
<b>Maybe</b>	<b>11,38</b>	<b>12,81</b>	<b>12,12</b>
hard	11,38	12,81	12,10
very hard	12,16	12,16	12,16
<b>No!</b>	<b>12,67</b>	<b>12,67</b>	<b>12,67</b>
very hard	12,67	12,67	12,67
<b>Yes, I enjoyed it</b>	<b>8,30</b>	<b>15,18</b>	<b>11,83</b>
easy	10,99	13,32	12,37
hard	8,30	15,18	11,75
very easy	10,78	12,74	11,84
very hard	8,82	14,11	11,66
<b>Yes, it was interesting</b>	<b>9,40</b>	<b>14,45</b>	<b>11,68</b>
hard	9,40	14,45	11,76
very hard	11,27	11,47	11,37
<b>DS</b>	<b>10,63</b>	<b>17,61</b>	<b>14,41</b>
<b>Maybe</b>	<b>12,67</b>	<b>14,30</b>	<b>13,48</b>
hard	12,67	14,30	13,48
<b>No!</b>	<b>14,48</b>	<b>14,93</b>	<b>14,71</b>
easy	14,48	14,93	14,71
<b>Yes, I enjoyed it</b>	<b>10,63</b>	<b>16,58</b>	<b>14,07</b>
easy	12,31	12,31	12,31
hard	10,63	16,58	14,28
<b>Yes, it was interesting</b>	<b>11,19</b>	<b>17,61</b>	<b>14,66</b>
easy	12,93	17,61	15,22
hard	11,93	17,32	14,54
very hard	11,19	11,19	11,19
<b>EGSI</b>	<b>8,81</b>	<b>18,43</b>	<b>14,56</b>
<b>Maybe</b>	<b>9,78</b>	<b>17,03</b>	<b>13,92</b>
easy	9,78	17,03	14,07
hard	10,36	16,79	13,72
very easy	13,26	16,94	15,87
very hard	11,51	14,55	12,84
<b>No!</b>	<b>8,96</b>	<b>17,32</b>	<b>13,09</b>
easy	10,10	17,32	13,53
hard	9,07	16,41	12,96
very easy	15,53	15,53	15,53
very hard	8,96	16,82	12,93
<b>Yes, I enjoyed it</b>	<b>8,81</b>	<b>17,93</b>	<b>15,08</b>
easy	9,72	17,89	15,27
hard	10,71	17,93	14,61
very easy	14,65	17,47	16,66
very hard	8,81	17,26	14,30
<b>Yes, it was interesting</b>	<b>8,83</b>	<b>18,43</b>	<b>14,64</b>
easy	8,83	18,26	15,05
hard	9,29	17,62	13,95
very easy	16,71	18,43	17,39
very hard	11,86	17,32	15,00

Row Labels	Min of RESULT	Max of RESULT	Average of RESULT
<b>CIVIL</b>	<b>8,30</b>	<b>15,18</b>	<b>11,84</b>
<b>DS</b>	<b>10,63</b>	<b>17,61</b>	<b>14,41</b>
<b>EGSI</b>	<b>8,81</b>	<b>18,43</b>	<b>14,56</b>
<b>ESTAP</b>	<b>8,33</b>	<b>15,37</b>	<b>12,63</b>
<b>Maybe</b>	<b>12,63</b>	<b>12,63</b>	<b>12,63</b>
hard	12,63	12,63	12,63
<b>No!</b>	<b>13,31</b>	<b>13,31</b>	<b>13,31</b>
very easy	13,31	13,31	13,31
<b>Yes, I enjoyed it</b>	<b>10,62</b>	<b>15,37</b>	<b>12,91</b>
easy	14,45	15,37	14,91
hard	10,74	14,70	12,43
very easy	14,15	14,15	14,15
very hard	10,62	12,97	12,13
<b>Yes, it was interesting</b>	<b>8,33</b>	<b>14,34</b>	<b>12,29</b>
easy	11,81	14,34	12,90
hard	8,33	13,80	11,39
very hard	13,35	14,21	13,78
<b>Textile</b>	<b>10,18</b>	<b>18,78</b>	<b>12,86</b>
<b>Maybe</b>	<b>14,56</b>	<b>14,56</b>	<b>14,56</b>
very hard	14,56	14,56	14,56
<b>No!</b>	<b>10,64</b>	<b>10,64</b>	<b>10,64</b>
hard	10,64	10,64	10,64
<b>Yes, I enjoyed it</b>	<b>10,37</b>	<b>18,78</b>	<b>13,72</b>
easy	13,43	18,78	16,10
hard	13,18	13,18	13,18
very hard	10,37	12,84	11,60
<b>Yes, it was interesting</b>	<b>10,18</b>	<b>14,25</b>	<b>12,10</b>
hard	11,18	12,07	11,62
very hard	10,18	14,25	12,43
<b>Grand Total</b>	<b>8,30</b>	<b>18,78</b>	<b>14,22</b>

■ **Figure 9** Students grades by opinions.

## 6 Discussion

The case study results provide valuable insights into the use of Kahoot! as a Student Response System for testing in higher education. In the interpretation phase, it was observed that these findings could inform the development of new and innovative approaches for assessment and evaluation in this context, contributing to ongoing efforts to improve the quality of education and student learning outcomes.

Naturally, non-technical courses, such as DS, ESTAP, and Textile, rated the assessment exercise as challenging (hard or very hard), which is understandable given their knowledge base in non-computing subjects. It ranged from 63.33% at DS and textile at 85.71%. The students still approved the assessment exercise as a valid test mechanism. Interestingly, EGSI students had varying opinions, with 42.50% rating the assessment exercise as easy and 42.75% rating it as hard.

Furthermore, students' opinions about the difficulty of the AE had little impact on their results. For instance, some students who found it hard achieved good results, whereas some who found it easy achieved lower results. However, better results were achieved by students who approved of this approach.

Most students highlighted the importance of explaining each question after their end. According to them, it allowed them to understand what they failed and improve their understanding of the subject matter.

Overall, the study's results were very positive, with 97.04% of the students achieving a grade higher than ten and only 4 out of 507 failed. Globally, the students approved the assessment exercise, with a 78.70% approval rating, and only 8.68% of participants disliked it.

## 7 Conclusion

The study demonstrates the effectiveness of Kahoot! as an SRS for assessment exercises in higher education. The use of Kahoot! resulted in high levels of student engagement, motivation, and learning outcomes. The majority of students achieved a grade higher than 10, with only 4 out of 507 failing. Additionally, 78.70% of participants approved the approach, while only 8.68% disapproved. This high approval rate suggests that students found the use of Kahoot! to be a valuable and effective tool for testing in higher education.

The results of this study have important implications for the development of new and innovative approaches to assessment and evaluation in higher education. Kahoot! and other gamification platforms have the potential to transform traditional methods of assessment, promoting active learning and engagement. The findings of this study also highlight the importance of providing students with real-time feedback and explaining the correct answers after the assessment exercises, which can improve their understanding and performance.

By incorporating interactive and engaging assessment exercises like Kahoot! into their teaching practices, instructors can promote active learning and provide students with a more engaging and rewarding educational experience. The study can help professors interested in adopting a game-based learning platform in their teaching practices.

This experiment will continue in the future, and distinct types of questions will be added to further explore the potential of Kahoot! and other gamification platforms in higher education assessment. Further, this approach will also be compared with other existing and having the same goal and explored using different tools. Finally, the team will explore the possibility of comparing results using digital with non-digital methods.

## References

- 1 Ibrahim S Al-Busaidi, Abdulhadi A Alamodi, Majid S Al-Musalhi, and Marwan A Al-Lamki. Using kahoot! as a formative assessment tool in medical education. *Journal of Taibah University Medical Sciences*, 13(4):349–353, 2018.
- 2 Azza Alawadhi and Emad AS Abu-Ayyash. Students’ perceptions of kahoot!: An exploratory mixed-method study in efl undergraduate classrooms in the uae. *Education and Information Technologies*, pages 1–30, 2021.
- 3 Pamela Baxter and Susan Jack. Qualitative case study methodology: Study design and implementation for novice researchers. *The qualitative report*, 13(4):544–559, 2010.
- 4 Paul Black and Dylan Wiliam. Assessment in education: Principles, policy & practice. *Assessment in Education: Principles, Policy & Practice*, 5(1):7–74, 1998.
- 5 Paul Black and Dylan Wiliam. Inside the black box: Raising standards through classroom assessment. *Phi delta kappan*, 80(2):139–148, 1998.
- 6 Eoghan Brady and Linda B Holcomb. Using kahoot! in the classroom to create engagement and active learning: A game-based technology solution for elearning novices. *Management Teaching Review*, 2(2):77–85, 2017. doi:10.1177/2379298116689783.
- 7 Jane E Caldwell. Clickers in the large classroom: Current research and best-practice tips. *CBE—Life Sciences Education*, 6(1):9–20, 2007.
- 8 Hui-Hua Chiang. Kahoot! in an efl reading class. *Journal of Language Teaching and Research*, 11(1):33–44, 2020.
- 9 Ian Clark. Formative assessment: Assessment is for self-regulated learning. *Educational psychology review*, 24:205–249, 2012.
- 10 Isabel Cortés-Pérez, Nicolás Zagalaz-Anula, María del Carmen López-Ruiz, Ángel Díaz-Fernández, Esther Obrero-Gaitán, and María del Carmen Osuna-Pérez. Study based on gamification of tests through kahoot!™ and reward game cards as an innovative tool in physiotherapy students: A preliminary study. *Healthcare*, 11(4):578, 2023. doi:10.3390/healthcare11040578.
- 11 Kathleen M Eisenhardt. Building theories from case study research. *The Academy of Management Review*, 14(4):532–550, 1989.
- 12 Scott Freeman, Sarah L Eddy, Miles McDonough, Michelle K Smith, Nnadozie Okoroafor, Hannah Jordt, and Mary Pat Wenderoth. Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences*, 111(23):8410–8415, 2014.
- 13 Derya Orhan Göksün and Gülden Gürsoy. Comparing success and engagement in gamified learning experiences via kahoot and quizizz. *Computers & Education*, 135, March 2019. doi:10.1016/j.compedu.2019.02.015.
- 14 Nathaniel J Hunsu, Olusola Adesope, and Dan James Bayly. A meta-analysis of the effects of audience response systems (clicker-based technologies) on cognition and affect. *Computers & Education*, 94:102–119, 2016.
- 15 Muhd Al-Aarifin Ismail and Jamilah Al-Muhammady Mohammad. Kahoot: A promising tool for formative assessment in medical education. *Education in medicine journal*, 9(2), 2017.
- 16 Kyoko Johns. Engaging and assessing students with technology: a review of kahoot! *Delta Kappa Gamma Bulletin*, 81(4):89, 2015.
- 17 Filipe Portela. Techteach – An innovative method to increase the students’ engagement at classrooms. *Information*, 11(10), 2020. doi:10.3390/info11100483.
- 18 Filipe Portela. Towards an engaging and gamified online learning environment – A real casestudy. *Information*, 13(2), 2022. doi:10.3390/info13020080.
- 19 Dylan Wiliam. What is assessment for learning? *Studies in Educational Evaluation*, 37(1):3–14, 2011.
- 20 Robert K Yin. *Case study research and applications: Design and methods*. SAGE Publications, 2017.

# NLP/AI Based Techniques for Programming Exercises Generation

Tiago Carvalho Freitas ✉

ALGORITMI Research Centre/LASI, University of Minho, Braga, Portugal

Alvaro Costa Neto ✉ 

Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Barretos, Brazil

Maria João Varanda Pereira ✉ 

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

Pedro Rangel Henriques ✉ 

ALGORITMI Research Centre/LASI, University of Minho, Braga, Portugal

---

## Abstract

This paper focuses on the enhancement of computer programming exercises generation to the benefit of both students and teachers. By exploring Natural Language Processing (NLP) and Machine Learning (ML) methods for automatic generation of text and source code, it is possible to semi-automatically construct programming exercises, aiding teachers to reduce redundant work and more easily apply active learning methodologies. This would not only allow them to still play a leading role in the teaching-learning process, but also provide students a better and more interactive learning experience. If embedded in a widely accessible website, an exercises generator with these Artificial Intelligence (AI) methods might be used directly by students, in order to obtain randomised lists of exercises for their own study, at their own time. The emergence of new and increasingly powerful technologies, such as the ones utilised by ChatGPT, raises the discussion about their use for exercise generation. Albeit highly capable, monetary and computational costs are still obstacles for wider adoption, as well as the possibility of incorrect results. This paper describes the characteristics and behaviour of several ML models applied and trained for text and code generation and their use to generate computer programming exercises. Finally, an analysis based on correctness and coherence of the resulting exercise statements and complementary source codes generated/produced is presented, and the role that this type of technology can play in a programming exercise automatic generation system is discussed.

**2012 ACM Subject Classification** Social and professional topics → Computer science education; Software and its engineering → Imperative languages; Computing methodologies → Machine learning; Software and its engineering → Parsers

**Keywords and phrases** Natural Language Processing, Computer Programming Education, Exercises Generation, Text Generation, Code Generation

**Digital Object Identifier** 10.4230/OASICS.ICPEC.2023.9

**Funding** This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Units Project Scope: UIDB/00319/2020 and UIDB/05757/2020.

## 1 Introduction

With technological advancements, computer programming education has become increasingly important in recent years, and along with it, the need for effective methods of teaching programming languages has become a priority.

A large part of the students have a lot of difficulties and consequent low approval rates in introductory computer programming courses, much because of the lack of motivation and attention inside and outside the classes [21]. To overcome those difficulties, students need to



© Tiago Carvalho Freitas, Alvaro Costa Neto, Maria João Varanda Pereira, and Pedro Rangel Henriques;

licensed under Creative Commons License CC-BY 4.0

4th International Computer Programming Education Conference (ICPEC 2023).

Editors: Ricardo Alexandre Peixoto de Queirós and Mário Paulo Teixeira Pinto; Article No. 9; pp. 9:1–9:12

OpenAccess Series in Informatics



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

practice frequently solving problems with an increasing complexity. On account of that, one of the major challenges faced by teachers is creating a large number of unique programming exercises that address different levels of difficulty. One possible way to help solve these issues is the development of a system for automatic generation of programming exercises based on Natural Language Processing (NLP) and Artificial Intelligence (AI) techniques capable of generating text and source code. In this paper we present and discuss the preliminary results of a research project focused on the analysis of ML models for text and code generation, aiming at building a generator for programming exercises.

This paper starts with a review at how programming exercises are usually constructed (Section 2), followed by a discussion on modern approaches for text and code generation (Section 3). Then, case studies regarding different generation mechanisms are presented, with the subsequent analysis and discussion of their results (Section 4). Finally, it is presented a summary of the paper and our main contribution, as well as a proposal of a road map for the future construction of a website that automatically generates computer programming exercises (Section 5).

## 2 Classification and Characterisation of Programming Exercises

Programming exercises are the key object of study for this project. It is then essential to gather the most used formats and understand their common characteristics to group them by type, aiming to automatically create them. In this way it is possible to avoid creating exercises that are unfamiliar to students and teachers.

### 2.1 Exercise Types

Listed below are some of the most common types of exercises in programming courses, gathered and adapted from the study published in [28]. It is important to note that, although there are additional taxonomies and categories, such as the ones proposed in [4, 26, 23], the former was chosen as it better resembles the structural nature of the exercises to be generated. The purpose in gathering such a variety of types is to present different challenges to students, on several themes. While some types are frequently used in open-ended questions, each one of these types can be adapted to multiple-choice formats, where students must select the correct answer from a list of options. This adaption is not trivial though, since it is necessary to generate not only the correct option, but also a set of incorrect ones.

Seven types were collected:

1. **Code from Scratch:** Students receive a blank sheet to write down the complete solution to a problem. Something that is easy for teachers to prepare, but also presents quite a lot of difficulties to those who are being evaluated. Listing 1 contains an example of a simple exercise that asks to write a function in Python.

■ **Listing 1** Example of *Code from Scratch* type exercises.

```
Write a program in Python that reverses a string.
Input: "cool"
Output: "looc"
```

2. **Code Completion:** To solve this type of problem the students are provided with a sample code containing some blanks to fill. There are usually two approaches: one in which an important part of the code is missing and it is necessary to fill it with the correct statements; and another whose blanks must be filled with values for which the program returns a given output. Listing 2 contains an example in which it is necessary to complete the condition in a selection statement in the body of a certain function in Python whose purpose is described.

■ **Listing 2** Example of *Code Completion* type exercise.

Complete the following Python function that receives a list of numbers and returns another one only with the even numbers.

```
def even(input_list):
    even_list = []
    for n in input_list:
        if _____:
            even_list.add(n)
    return even_list
```

3. **Code Improvement:** Another approach consists in providing students a complete snippet of code that solves a given problem which should be improved. These improvements may have as objective improving the performance or reducing lines of code. Listing 3 contains an exercise in which it is proposed to rewrite a function in Python to a reduced version.

■ **Listing 3** Example of *Code Improvement* type exercises.

Rewrite the following code snippet in order to make it smaller, maintaining its functionality.

```
my_list = list(range(1,100))
res = 0
i = 0
while i < len(my_list):
    if my_list[i] % 3 == 0:
        if my_list[i] % 2 == 1:
            res += my_list[i]
        i += 1
return res
```

4. **Bug Finding:** As the name suggests, in this type of exercise the students must find in code purposefully added bugs without correcting them *per se*. This type of exercise is used to check if students attained the first step of learning to program, that is to be capable of interpreting the code written by others in terms of the syntax or semantics of the language used. Listing 4 contains an exercise in which it is asked to identify bugs in a function and explain in a few lines why they are bugs.

■ **Listing 4** Example of *Bug Finding* type exercises.

Identify the bug(s) present in the following code sample.

```
def my_function(n):
    a = 1
    my_list = [1,2,3]
    if n > 3:
        return n + b
    else:
        return a + my_list[3]
```

5. **Debugging and Fixing:** Interestingly, if the two previous types of exercises are mixed together, the result is also quite common, in which students must rewrite a source code that contains bugs. It is necessary not only to detect them, but also to fix the code.
6. **Code Interpretation:** As the name suggests, in this type of exercise the student has to interpret a given code snippet. There are many possibilities for what is asked, such as: describing its behaviour, identifying its goals, reporting the evolution of a variable's value throughout its execution, etc. Listing 5 contains an exercise in which it is asked to describe what a given function does.

■ **Listing 5** Example of *Code Interpretation* type exercises.

Describe the behaviour and the goal of the following function.

```
def my_function(str):
    if len(str) < 2:
        return ''
    return str[0:2] + str[-2:]
```

7. **Output or State Prediction:** Finally, and in a more traditional format, the students are asked to find out either the output of the program, or the value of a variable at the end or during its run-time. This ends up being very similar to what was discussed in the last item, as it is necessary to analyse and interpret the provided code to obtain the correct solution. Listing 6 contains an exercise where it is asked to indicate what value a function call returns.

■ **Listing 6** Example of *Output or State Prediction* type exercises.

```
Indicate what the following function returns as an output.

def my_function(n):
    my_list = []
    for i in range(n):
        my_list.append(i*i)
    return my_list

my_function(3)
```

## 2.2 Exercise Components

It is important to understand the components that make up a programming exercise in order to effectively design and create them. There are usually three main components that need to be considered:

- **Problem Statement:** Initial component containing text that is presented to the student. It explains the context and parameters of the problem, and the type of answer they must provide. It can be lengthy if the topic needs an introduction or concise and objective if it only requires the essential instructions for solving the problem. It is important to provide enough information to solve the problem, specially when the domain is unknown to the student.
- **Code:** Most programming exercises contain a section dedicated to code snippets. This section may vary in form, containing whole programs, snippets with blanks to be filled, and correct or erroneous versions to be analysed or fixed. The purpose of this component is to support the *Problem Statement* to establish a basis for solving the problem.
- **Answer field or options:** There are two usual possibilities for the answer section, either a designated space or an area embedded in another component. The former includes a blank space where students are free to write whatever they want, or a list of available options to choose from. The latter usually consists of blank spaces embedded in the *Code* component, visually identifying where the students must fill in their answers.

While not a component in itself, the evaluation of the students' answers must be taken into account. Its complexity may vary depending on the type of the exercise, ranging from a direct verification of the correct option, to more sophisticated mechanisms, such as a set of tests performed on the code that the student has written. Although, some characteristics of the answers present a qualitative nature that prevents its automatic evaluation, such as code sophistication and legibility.

In summary, understanding the components of a programming exercise is crucial to design effective exercises. These components can vary depending on the type of exercise and its objectives.

## 3 Text Generation

Natural Language Generation (NLG) is a sub-field of Natural Language Processing (NLP) that involves building systems capable of producing coherent and useful text in multiple languages [24]. NLG systems are used to create chatbots, translate text, and generate

complex articles and stories [6]. This process involves taking input data, such as keywords or a set of facts, and transforming them into meaningful output text. Reiter and Dale [25] identified six tasks that are essential for NLG, still present in current models:

1. **Content Determination:** Deciding what information should be included in the text under construction;
2. **Discourse Planning or Text Structuring:** Defining in which order information will be presented and the structure of such presentation;
3. **Sentence Aggregation:** Grouping information into sentences;
4. **Lexicalization:** Deciding which words and phrases should be chosen to express the required information;
5. **Referring Expression Generation:** According to Reiter and Dale [25], it consists of “selecting the words and phrases to identify domain objects”;
6. **Linguistic Realisation:** Combining all words and phrases by applying grammatical rules to produce a text which is syntactically and orthographically correct.

This type of system can be built using different architectures, being the most common the three-stage pipeline, also proposed by Reiter and Dale [24]. The pipeline includes text planning, sentence planning, and linguistic realisation. In text planning, content determination and discourse planning are combined to decide what information to include and how to structure it. In sentence planning, sentence aggregation, lexicalization, and referring expression generation are combined to select words and phrases and group them into sentences. Finally, in linguistic realisation, grammatical rules are applied to produce a syntactically and orthographically correct text.

### 3.1 Models

Natural Language Processing (NLP) is a sub-field of Artificial Intelligence (AI) that focuses on the interaction between computers and human language [13]. Machine Learning algorithms have been developed and trained using text data to perform various natural language tasks such as text prediction and generation [17]. This section will explore some of the most commonly used and successful NLP models, as well as other generation systems.

#### Recurrent Neural Networks (RNNs)

Neural Networks are a subset of Machine Learning that are inspired by the structure and function of the human brain. They are used to classify and cluster data, and consist of layers of artificial neurons that send data to each other. The output of these neurons is determined by weights and threshold values, and training data is used to improve their accuracy over time [11].

One type of neural network architecture that is well-suited for processing sequential data, such as text, is Recurrent Neural Networks (RNNs). RNNs have feedback connections that allow them to incorporate information from previous time steps or observations into their current output. This makes them effective at modelling dependencies between elements in a sequence to predict the next one [12].

- **Long Short-Term Memory (LSTM) networks:** A type of RNN that is particularly well-suited for modelling long-term dependencies in sequential data, such as natural language text. LSTM networks are able to “remember” important information from the past for extended periods of time, by using state cells, making them ideal for generating coherent and realistic text [9]. For example, in the paper [15], the authors trained a LSTM on a dataset of Shakespearean plays was able to generate text that was difficult to distinguish from human-written equivalents.

- **Seq2seq models:** Have been widely used for NLP tasks that involve the generation of an output sequence based on an input sequence. Seq2seq models consist of an encoder and a decoder (each one a model with neural network architecture), which process the input and output sequences. One of their key advantages is the ability to handle variable-length input and output sequences [8]. Seq2seq models can be implemented using various types of neural network architectures, although RNNs are the most common. For example, in the article [29], the authors use a Seq2seq model with an RNN architecture to generate translations from one language to another.

In conclusion, Recurrent Neural Networks, such as Long Short-Term Memory networks and Seq2seq models are among the most commonly used and successful NLP models at the moment. They are able to effectively process sequential data, such as natural language text, and generate coherent and realistic output.

## Transformers

Transformers consist in a type of neural network architecture that was first described in an article by Vaswani et al. [30], which has been effective in NLP tasks. The capacity of transformers to model long-range dependencies between words in a sentence or text is one of its most important features. This is done by utilising self-attention mechanisms, which give the network the ability to weigh the importance of the different words or tokens in the input based on how they relate to other words or tokens. Transformers are thus well suited for tasks that require a thorough comprehension of the context and semantics of a particular text. Some of the most popular and successful transformer models are:

- **Generative Pre-training Transformer (GPT):** Transformer-based language model developed by OpenAI [19], that has been widely used for natural language processing tasks such as text generation. It uses unsupervised learning to pre-train a large neural network on a massive dataset of text, discovering “hidden patterns or data groupings without the need for human intervention” [14], and then fine-tunes the network on a smaller dataset of labeled text for a specific task. GPT-3 is the third generation of GPT models, which was introduced in 2020 [5] and. This model are significantly larger and more powerful than its predecessor models, since it is trained on 175 billion parameters (over ten times the size of GPT-2) and requires very little fine-tuning on a task-specific dataset to achieve good performance on a wide range of NLP tasks. OpenAI also launched GPT-3.5, an upgrade for this model, and a new version, GPT-4 [20], which is even more powerful (100 trillion parameters). ChatGPT is a well-known application of GPT that is specifically designed for conversational language understanding and generation, trained on conversational data [18].

It is worth noting that, as GPT-3 is an unsupervised model, it is vulnerable to biases present in its training dataset, and may generate texts that reflect them.

- **BERT:** Transformer-based language model that was developed by Google in 2018 [7] and, in similar form to GPT, it also uses unsupervised learning to train a neural network on a large dataset of text. The main difference between GPT and BERT is that BERT is a masked language model, trained to predict missing words in a sentence rather than the next word, as GPT does. This makes the text generated by BERT less likely to diverge from the original context of the input. In one study, researchers used BERT to generate news articles that were evaluated by human annotators, who found them comparable in quality to those written by humans [34].

Source:

```
prompt = "Create a programming exercise in Python about string
manipulation and write the answer."
exercise = openai.Completion.create(engine="text-davinci-003",
prompt=prompt, temperature=0.7, max_tokens=500, top_p=1,
frequency_penalty=0, presence_penalty=0)
```

Output:

```
Write a program to print out the last three characters of a string.

#Answer:
string = input("Enter a string: ")
length = len(string)
last_three_chars = string[length - 3:]
print(last_three_chars)
```

■ **Figure 1** Example of a programming exercise generation using GPT-3.5.

## 4 Case Studies and Results

This section introduces some case studies to explore different implementations of up-to-date text generation models. The outcomes are then analysed in order to assess some of the most commonly used mechanisms and how they can be improved for the generation of programming exercises.

### 4.1 OpenAI API

OpenAI has made available APIs to GPT-3 and GPT-3.5 models. Using an access token it is possible to make requests by sending a prompt with a certain maximum number of tokens, which will be answered with the generated text. Each of these models has some variations: some are applied to simpler and faster tasks while others are used for more complex text generation. Despite its technical capacity and current popularity, there is an important tangential disadvantage: the API is not completely free. After a gracing period of three months, there are monetary usage fees for the different OpenAI models, which despite appearing minimal (GPT-3.5 is 0.002\$ per 1K tokens) may become a problem in an academic context with a lot of tests. Beyond that, for actual implementation in harsher socioeconomic contexts, it might become a dividing criteria that opposes the already slow-paced democratisation of science and education, while concurrently enlarging the technological gap between world-wide communities.

In the context of generating programming exercises, this tool presents satisfactory results. By making requests with the type of the exercise, theme or more concrete characteristics, such as the mandatory use of a language structure or construct, it is possible to obtain complete exercises, including their corresponding answers. The only disadvantage that was realised about the quality and effectiveness of the results is the possible incoherence derived from the lack of context of the model, that is, it may produce correct text, but outside the scope of the problem in question. To improve this, OpenAI also provides the option of fine-tuning, in which the model is further trained with a shorter dataset to perform a specific task. By fine-tuning GPT, it would be possible to obtain faster and more adequate results, in the sense that the exercises and their parts would be in accordance with a pre-defined structure and better contextualised with the domain in question.

Figure 1 shows the generation of a programming exercise from a prompt, in which the latest GPT-3.5 model was used (*text-davinci-003*). The model was configured using parameters recommended in the OpenAI documentation. As can be seen, the result for a

Source:

```
prompt = """Exercise

Instructions:
Write a Python program to reverse a string.

Code: """

model.generate(prompt=prompt, max_length=500, temperature=0.3, top_p=0.9)
```

Output:

```
Exercise

Instructions:
Write a Python program to reverse the binary representation of a given
integer.

Code:
def test(n):
    return int(bin(n)[::-1][:-2], 2)
```

■ **Figure 2** Example of a programming exercise generation using a finetuned GPT-2 model.

relatively small input is quite satisfactory and there are a lot of possible generations for the same input. The potential is clearly enormous with regard to the type, structure, subject and difficulty of the exercises.

## 4.2 Finetuning GPT-2

Among GPT models, it is possible to work with older versions for free. There are even libraries such as *aitextgen* [33] that provide the possibility to finetune the GPT-2 model from a dataset. Using PyTorch and TensorFlow (machine learning libraries), it is possible to improve the performance of a specific task within text generation. However, this version is not as powerful as the most recent ones, since it was trained with 100 times fewer parameters, which makes the results not as adequate. Furthermore, it does not provide as much flexibility in order to perform different text generation tasks (question answering, summarization, text classification and so on) [27]. The fact that there is no remote access to use and train GPT-2 makes it free, but also requires all processes to be done locally, demanding a significantly greater use of computational resources. Figure 2 displays the output of an attempt that was made to finetune GPT-2 from a dataset consisting of a list of 300 Python basic exercises taken from the W3 Resource website [31].

As GPT-2 is not prepared to answer requests, but to complete text, the dataset was modified in order to define the structure of the exercise as *Problem Statement* and *Code*. When analysing the result obtained for the provided prompt, it can be seen that the model was not able to generate according to the predefined *Problem Statement* and, with that, generated another text with some similarities, maintaining the structure, but incoherently and out of the context.

The quality of the results is not the one that is expected and this is mainly due to the small size of the dataset. The model to be trained with this dataset took about 1 hour, which implies that, with a dataset of adequate size, it would require spending a huge amount of time.

Source:

```
keystotext_model.predict(["delete", "list", "odd numbers"])
```

Output:

```
Write a function to delete odd numbers from a list.
```

■ **Figure 3** Example of a *Problem Statement* component generation using the trained *keys-to-text* model.

### 4.3 Key-to-Text

The *keytotext* library [2] was used in the attempt to generate the *Problem Statement* component of an exercise, which implements a model that receives keywords as input and generates sentences. It is based on the Text-to-Text Transfer Transformer (T5) model, a transformer-based language model developed by Google's AI team that employs sequence-to-sequence (Seq2Seq) pre-training and uses a fine-tuning approach to perform a variety of NLP tasks [22]. The T5 model is one of the largest transformer-based language models to date, with 11 billion parameters.

The *keytotext* library simplified the process of fine-tuning the original model to work with computing programming exercises generation. Using the Mostly Basic Python Problems (MBPP) dataset from Google Research [1], which consists of 1000 programming problems designed to be solvable by entry level programmers, the model was fine-tuned to perform in a contextualised and efficient way. In order to train the model to generate exercise *Problem Statement* from keywords, a field with the three most relevant keywords that represent the text of each exercise was included in the dataset. The NLTK platform [3] was used to accomplish this, since it has numerous libraries and functionalities for working with natural language, such as RAKE that can extract and sort words from a given text by importance.

By applying this method of text generation to the context of the programming exercises, it was possible to obtain satisfactory results. It should be noted that, as the model was trained with a trio of keywords for each text, it will lose its efficiency if provided with a different quantity of keywords.

As illustrated in Figure 3, the results are quite satisfactory and coherent within the context of the dataset provided. Sometimes, the generated text does not correspond to what the user expects, but it manages to make sense according to the words entered. This provides a wider variety of exercises helping in the creative process.

### 4.4 CodeT5

*CodeT5* [32] builds on the T5 architecture, but incorporates source code-specific knowledge to endow the model with better code understanding. It was trained on the CodeSearchNet dataset, which consists of millions of lines of code written in popular programming languages such as Python, Java, and JavaScript, as well as popular frameworks and libraries such as TensorFlow, React, and Django [10]. This enables the model to generate code that is not only syntactically but also functionally correct and efficient. It was tested using one of the checkpoints of this model (*CodeT5-large-ntp-py*) which was trained on millions of Python source code files from GitHub (GCPY - Python Github Code Dataset) [16].

The source shown in Figure 4 requested from the model a function in Python with substantial complexity, including string manipulation and the use of a repetition structure (loop). The model produced code that performs what was requested in an optimised way. However, being a somewhat computationally demanding mechanism, it required significant time to load and generate the corresponding code.

## 9:10 NLP/AI Based Techniques for Programming Exercises Generation

Source:

```
txt = "Function that counts the frequency of each character in a string."  
input_ids = tokenizer(txt, return_tensors="pt").input_ids  
generated_ids = model.generate(input_ids, max_length=128)  
print(tokenizer.decode(generated_ids[0], skip_special_tokens=True))
```

Output:

```
def count_characters(string):  
    char_freq = {}  
    for char in string:  
        char_freq[char] = char_freq.get(char, 0) + 1  
    return char_freq
```

■ **Figure 4** Example of code generation using *CodeT5*.

Despite using a constant string as input to the model in Figure 4 (variable `txt`, used for exemplification purposes only), the real test was conducted by linking the output of the *keytotext* model to the input of *CodeT5*, creating a pipeline that generated not only the *Problem Statement*, but also the *Code* for that specific exercise directly.

### 4.5 Summary

After the review and implementation of NLP/ML methods that can be integrated into a tool for automatic generation of programming exercises, it was possible to analyse various approaches, comparing advantages and disadvantages between them.

As drawbacks, the price may be seen as a barrier for public and free implementations, while computational resources required by some implementations, such as GPT-3, also incur in initial costs to infrastructure construction and maintenance. These factors are undoubtedly important in order to choose which approach to adopt for the implementation of an exercise generator. Another aspect that should obviously be discussed is the quality of the generated exercises, in terms of lexical, syntactic and semantic correctness and overall coherence to the input parameters.

The initial approach to generate a complete exercise (problem statement, code snippet and answer options), using theme and type as inputs, required an advanced model (GPT-3). This raised three problems:

- Implementing such a system on an ordinary machine is quite complicated, as it requires resources that common computers do not have, making this process very slow or even unattainable;
- The need to use costly APIs would present a financial strain that could be unsustainable in the medium or long term;
- Creating a system dependent on an Internet connection could lead to communication failures and reliability issues.

The approach of generating the components of an exercise separately has proven to be more efficient, as it lies upon the use of simpler models. The first component was generated by the *keytotext* library in tandem with the MBPP dataset, resulting in a fully constructed and comprehensible problem statement. In turn, this problem statement was then fed to the *CodeT5* model in order to generate the second component of the exercise, namely the source code snippet. Besides other models with higher complexity were tested, this approach – used to generate around 100 exercises – demonstrated equally capable results to GPT-3, with approximately 80% rate of success in generating comprehensible and coherent exercises.

## 5 Conclusion

The investigation and exploration reported in this paper contributed to establish a panorama on how to automatically generate computer programming exercises, using acceptable NLP and ML mechanisms to construct both the *Problem Statement* and its accompanying *Code*. A prototype for the system that implements these mechanisms has been developed and tested, in order to pave the way to the final semi-automatic exercise generator we intend to build.

Future works in this project include a compiler to a Domain Specific Language (DSL) that has been designed to facilitate the exercises generation process, templates to automatically create different versions of the *Problem Statement* and *Code*, and user interfaces to create new templates and retrieve lists of exercises. In that way teachers and students will benefit from our system, getting different and challenging problems to practice and test their programming abilities.

---

## References

- 1 Jacob Austin, Augustus Odena, Maxwell Nye, et al. Program synthesis with large language models. *arXiv preprint*, 2021. [arXiv:2108.07732](https://arxiv.org/abs/2108.07732).
- 2 Gagan Bhatia. keytotext. URL: <https://github.com/gagan3012/keytotext>.
- 3 Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.
- 4 Matt Bower. A taxonomy of task types in computing. *SIGCSE Bull.*, 40(3):281–285, June 2008. doi:10.1145/1597849.1384346.
- 5 Tom B. Brown, Benjamin Mann, Nick Ryder, et al. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. [arXiv:2005.14165](https://arxiv.org/abs/2005.14165).
- 6 Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. Evaluation of text generation: A survey, 2021. [arXiv:2006.14799](https://arxiv.org/abs/2006.14799).
- 7 Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi:10.18653/v1/n19-1423.
- 8 Pranay Dugar. Attention – Seq2Seq Models. <https://towardsdatascience.com/day-1-2-attention-seq2seq-models-65df3f49e263>, 2019.
- 9 Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. [arXiv:1308.0850](https://arxiv.org/abs/1308.0850).
- 10 Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. Codesearchnet challenge: Evaluating the state of semantic code search. *CoRR*, abs/1909.09436, 2019. [arXiv:1909.09436](https://arxiv.org/abs/1909.09436).
- 11 IBM. What are Neural Networks? <https://www.ibm.com/topics/neural-networks>, 2023.
- 12 IBM. What are Recurrent Neural Networks? <https://www.ibm.com/topics/recurrent-neural-networks>, 2023.
- 13 IBM. What is Natural Language Processing? <https://www.ibm.com/topics/natural-languageprocessing>, 2023.
- 14 IBM. What is Unsupervised Learning? <https://www.ibm.com/topics/unsupervised-learning>, 2023.
- 15 Andrej Karpathy. The unreasonable effectiveness of recurrentneural networks, 2015. URL: <http://karpathy.github.io/2015/05/21/rnneffectiveness/>.
- 16 Hung Le, Yue Wang, Akhilesh Gotmare, Silvio Savarese, and Steven Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning, July 2022. doi:10.48550/arXiv.2207.01780.

- 17 Archana Oberoi. What are Language Models in NLP? <https://insights.daffodilsw.com/blog/what-are-language-models-in-nlp>, 2020.
- 18 OpenAI. ChatGPT. <https://openai.com/blog/chatgpt>.
- 19 OpenAI. OpenAI. <https://www.openai.com/product>.
- 20 OpenAI. Gpt-4 technical report, 2023. [arXiv:2303.08774](https://arxiv.org/abs/2303.08774).
- 21 Mário Pinto and Teresa Terroso. Learning Computer Programming: A Gamified Approach. In Alberto Simões and João Carlos Silva, editors, *Third International Computer Programming Education Conference (ICPEC 2022)*, volume 102 of *Open Access Series in Informatics (OASICs)*, pages 11:1–11:8, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: <https://drops.dagstuhl.de/opus/volltexte/2022/16615>.
- 22 Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), January 2020.
- 23 Noa Ragonis. Type of questions – The case of computer science. *Olympiads in Informatics*, 6:115–132, January 2012.
- 24 Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, 2000.
- 25 Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Natural Language Engineering*, 3, March 2002.
- 26 Alexander Ruf, Marc Berges, and Peter Hubwieser. Classification of programming tasks according to required skills and knowledge representation. In *Informatics in Schools. Curricula, Competences, and Competitions - 8th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2015, Ljubljana, Slovenia, September 28 - October 1, 2015, Proceedings*, volume 9378, September 2015. doi:10.1007/978-3-319-25396-1\_6.
- 27 Gianetan Sekhon. Gpt-2 vs gpt-3. <https://medium.com/@gianetan/gpt-2-vs-gpt-3-e915ac43e981>, 2023.
- 28 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.
- 29 Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems – Volume 2, NIPS’14*, pages 3104–3112, Cambridge, MA, USA, 2014. MIT Press.
- 30 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17*, pages 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.
- 31 w3resource. Python exercises, practice, solution. <https://www.w3resource.com/python-exercises/>, 2023.
- 32 Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8696–8708, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL: <https://aclanthology.org/2021.emnlp-main.685>.
- 33 Max Woolf. aitextgen. <https://docs.aitextgen.io/>, 2021.
- 34 Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020.

# Data Visualization for Learning Analytics Indicators in Programming Teaching

Ranieri Alves dos Santos ✉ 

Graduate Program in Knowledge Engineering and Management, UFSC, Florianópolis, Brazil

Dalner Barbi ✉ 

Graduate Program in Knowledge Engineering and Management, UFSC, Florianópolis, Brazil

Vinicius Faria Culmant Ramos ✉ 

Graduate Program in Knowledge Engineering and Management, UFSC, Florianópolis, Brazil

Fernando Alvaro Ostuni Gauthier ✉ 

Graduate Program in Knowledge Engineering and Management, UFSC, Florianópolis, Brazil

---

## Abstract

Learning Analytics (LA) has the potential to transform the way we learn, work and live our lives. To reach its potential, it must be clearly defined, incorporated into institutional teaching-learning strategies and processes and practices. The main goal of this study is to list indicators to be used in learning analytics in programming teaching and how to expose their views. For the development of the indicator model, this study based on a qualitative analysis, using data visualization and business intelligence tools, in projects focused on Learning Analytics. As a result, four main indicators were mapped: accesses to the system, resources accessed, activities carried out and, performance in activities.

**2012 ACM Subject Classification** Information systems → Data analytics

**Keywords and phrases** learning analytics, data visualization, learning indicators

**Digital Object Identifier** 10.4230/OASICS.ICPEC.2023.10

**Category** Short Paper

**Funding** This work was carried out with the support of the Coordination for the Improvement of Higher Education Personnel – Brazil (CAPES) – Financing Code 001.

## 1 Introduction

In the context of the academic domain, Learning Analytics (LA) has grown rapidly, with a large volume of special editions of scientific journals in education, psychology, computing and social sciences focused on this topic. Due to its recent use, the organization of its fundamental concepts, theories, techniques, methods, approaches, applications and strategies has been structured for its development in several fields of action.

Since the emergence of digital teaching and learning platforms, which support educational processes, it has been possible to monitor the behavior of students and teachers in a broad and detailed way, offering a range of precision and prediction to the effectiveness of learning. And, in this context, new computational and interactional techniques are being created to absorb all the issues that involve it, from statistical results, purely, to issues such as ethics and privacy [17, 4, 11].

The field of Learning Analytics is attractive as a result of the possibility of using large volumes and a variety of data that enhance the vision of substantial improvements in teaching and learning practices, in an optimized and scalable way [9, 12].



© Ranieri Alves dos Santos, Dalner Barbi, Vinicius Faria Culmant Ramos, and Fernando Alvaro Ostuni Gauthier;

licensed under Creative Commons License CC-BY 4.0

4th International Computer Programming Education Conference (ICPEC 2023).

Editors: Ricardo Alexandre Peixoto de Queirós and Mário Paulo Teixeira Pinto; Article No. 10; pp. 10:1–10:7

OpenAccess Series in Informatics



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

## 10:2 Data Visualization for Learning Analytics Indicators in Programming Teaching

As Clow [4] exposes, the exponential availability of data on learning activities in digital environments, with an approach centered on quantitative metrics, contributes to the understanding of educational processes that, in the context of the analysis, offers a broad view for teachers to use its resources, conceptions and teaching methods more effectively, providing better results.

Indeed, the field of Learning Analytics has been developing in a society permeated by the power of algorithms, mathematical approaches, advanced conditions of technological information processing equipment and data analysis methods, impacting education management and the process assessment as a foundation for the epistemological assumptions of pedagogical practices [8, 6].

### 2 Indicators for Learning Analytics

As the field of Learning Analytics is very recent, indicators of learning behavior in virtual learning environments with some degree of standardization have not yet been determined, to be used in the development of learning analyzes [12, 16].

Using a group concept mapping (GCM) approach, a tool available online, [16] create five dimensions of qualitative indicators for use in Learning Analytics assessments: a) objectives (awareness, reflection, motivation, behavioral change); b) learning support (perceived usefulness, recommendation, classification of activities, detection of students at risk); c) measures of learning and outcomes (comparability, effectiveness, efficiency, usefulness); d) data aspects (transparency, data standards, data ownership, privacy); and, e) organizational aspects (availability, implementation, training of educational stakeholders, organizational change).

In another study, Maraza-Quispe et al. [12] use a quantitative approach to propose indicators of learning behavior in virtual environments, to efficiently develop analytical learning processes, aiming at more effective predictions about learners' performance, decision-making and optimization of learning processes. They are: a) preparation for learning; b) progress along the course; c) learning resources; d) interaction in the forums; and, e) assessment of resources.

### 3 Data Visualization

Data visualization aims to provide ways to graphically represent collections of data. There are several ways to visualize and represent data, which go back to the first forms of information representation, such as maps, graphs and other graphics[18]. Data visualization technology acts as a cognitive aid to understand what you want to report, and it is a device that allows you to visualize nuances laid out only in the form of raw data in an understandable way. In this way, these techniques facilitate research, add meaning, capture information, collaborative discoveries and assist in the process of discovery and identification of elements.

Although clarity, objectivity and the ability to allow logical reasoning between data represent an aspect of data visualization, the reliability of the database and the reality it represents has a clear value relationship with visualization capabilities [13, 2, 3]. However, the data cannot speak for themselves, requiring the interlocutors to articulate variables for them in the discourse, from which the expected inferences of the reader can be generated. This work, sometimes attributed to data journalists, sometimes to researchers, condenses, aggregates, cuts or adjusts data so that it can be analyzed and disseminated. According to

his ongoing project, with the advent of big data and the democratization of data visualization, it has become essential that executives and business analysts have the power to choose the information that is relevant to them [18].

The neutrality of the graphical representation agents as authors of the information and the neutrality of the elements that previously develop data acquisition patterns and variables are usually analyzed as much as the information provided by these agents. Therefore, just as the possibility of improving the comprehensibility of quantitative information through cognitive sources is widely discussed, it is essential to discuss the magnitude of biases that can be aggregated within the same source of information and the aspects that affect its reliability in projects of data visualization [2].

New ways of presenting data make it possible to instantly find insights that can now be displayed anywhere in the world. Thus, new data visualization tools and techniques address an old problem where knowledge is seen as synonymous with power, but having too much information can be counterproductive [18].

## 4 Methodological Procedures

For the development of the indicators model, this work was based on a qualitative analysis of approaches that use data visualization and business intelligence in projects focused on Learning Analytics. Therefore, a scope review was carried out using the acronym PCC (Population, Concept and Context).

The population was stipulated as being “indicators in Learning Analytics”, the concept as “data visualization techniques” and the context as “technologies in higher education”. As a research period, references published between 2012 and 2023 were sought, performing the search with the string “learning analytics” AND (“data visualization” OR “indicators”). The research sought to integrate several sources of publications about educational technology, for which the search used the LearnTechLib, Google Scholar and Web of Science databases and resulted in 1655 records.

Considering the only exclusion criteria is articles not related to the research question of this work, an exploratory qualitative research was carried out, focusing on the mapping of initiatives in Learning Analytics and data visualization. All articles that did not present relevant proposals on the use of visualization of learning analytics data and that were applicable to the teaching environment of programming in e-learning were excluded, in the end, 7 articles were selected.

## 5 Results and Discussions

From the works resulting from the review, Ali et al. [1] propose a tool for visualizing qualitative data in learning analytics, Essa and Ayad [5] propose a student success analysis system based on the visualization and prediction of learning data analytics, meanwhile, Khuzairi et al. [7] reviews the use of data visualization in learning analytics in the literature, Paiva et al. [14] proposes data mining in learning analytics, with data visualization, Phillips et al. [15] develops a classification algorithm for student engagement data in learning analytics, Song [19] analyzes data mining and visualization techniques in the context of learning analytics, Uskov et al. [20] works on the analysis of learning in higher education using data cleaning and visualization techniques.

The indicators to compose the dashboard proposed in this work were based on the works of Khuzairi et al. [7], Phillips et al. [15], Essa and Ayad [5] and Ali et al. [1]. These works served as a reference for defining the variables that will serve as a basis for viewing and

## 10:4 Data Visualization for Learning Analytics Indicators in Programming Teaching

classifying the educational data that will be displayed on the dashboard. Based on the works by Song [19], Paiva et al. [14] and Khuzairi et al. [7], data visualization strategies and navigation modes between data contexts (data drilling) were developed.

As a result of the research, four main indicators were mapped, raised in the literature that would be able to provide visualizations that would facilitate the process of learning analytics in programming teaching. Accesses to the System: indicator that measures the number of accesses to the system that the user made, regardless of the actions carried out in the educational system, only their accesses.

Resources Accessed: indicator referring to the amount of resources accessed within the educational system, compared to the amount of resources available.

Performed Activities: indicator that measures the amount of activities carried out in the system, comparing with the amount of activities available, without taking into account their mistakes and successes, only the performance.

Performance in Activities: indicator that measures errors and successes in activities available in the system.

The indicators of “Accesses to the System”, “Resources Accessed” and “Performed Activities” are important to measure the performance of programming students because, according to Phillips et al[15], students with too much access to educational resources tend to be very effective in balancing your study commitments, contributing to your performance. In this way, on the contrary, when the student does not access the educational resources on the e-learning platform, his programming learning is compromised. The act of accessing or not accessing educational resources and the platform is not the main indicator of success or failure of the programming student. For Macarini et al[10], for example, counting any student interaction with the platform may be enough to predict early failure of programming students.

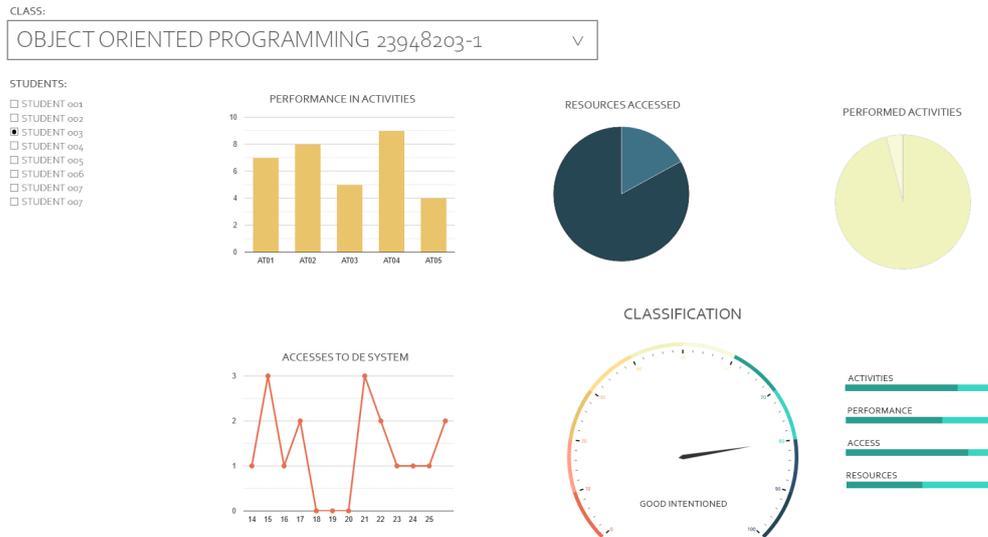
However, it is an indicator that contributes to the visualization of the learning analytics data about the student’s commitment. “Performance in Activities” is the main indicator of student performance success to be visualized. This indicator, along with the three other indicators, make up the gauge visualization strategy in the proposed dashboard.

Based on the mapped indicators, a proposal for data visualization of these indicators was developed based on the “BI Dashboard” visualization style, with contextual visualizations, enabling data drilling by class and by students. Visualizations will be plotted in bar, pie and line graphs (Figure 1).

On the upper left side, the first data drill option appears, where the user can access views by class, in the example of Figure 1, the performance of the indicators in the class of “Object-Oriented Programming – 23948203-1” is displayed. The teacher will be able to browse the information of their classes and the educational manager will be able to access the data of all classes.

Just below, on the left side, the second data drill option is displayed, which allows the teacher to choose which student he would like to visualize his performance. The teacher or manager can group one or more students to obtain their average performance.

In the visualization graphs, it will be possible to observe the student’s performance in the activities carried out, where from a bar graph, their assertiveness can be measured against the expected context. In the next pie chart, the teacher will be able to easily visualize the percentage of available resources that the user has accessed, the same happens with the next pie chart, which visualizes the percentage of activities carried out by the student. Just below, the line graph displays the number of accesses to the system in the historical series and a gauge with the student’s classification.



■ **Figure 1** Overview of the Proposed Model Dashboard.

The student rating scale is displayed in a gauge at the lower right corner (Figure 2). The gauge is based on the algorithm by Phillips et al [15], which uses the four indicators mapped in this work to specify the student’s level of involvement within the course. This algorithm, based on the indicators, classifies the student in ascending order in the following terms: non-user, random, single-use, accidental, free-time, inserted, constant user, well-intentioned, engaged and conscious, the latter being the student with the highest ranking against the algorithm.



■ **Figure 2** Overview of the Proposed Model Dashboard.

## 6 Conclusion

With the development of the proposed Learning Analytics data visualization model applied to programming teaching, it was possible to observe that the literature presents several approaches that facilitate the development of new approaches for the analysis of educational data from the actions already carried out and mapped fundamentally in online educational and learning management systems.

In this way, the present work presented a form of data visualization based on existing approaches in the literature capable of facilitating the analysis of the performance of classes and of specific students from contextual visualizations with data drilling navigation.

Initiatives aimed at simplifying quantitative symbologies from raw data in infographics, which are capable of composing strategies that facilitate the understanding of information for the appropriate target audience, can be of great contribution to the most diverse sectors, especially in education. Teachers and educational managers having access to simplified graphical views, which allow a quick understanding of the current status of certain students or classes, can dedicate more time to other activities, since they will no longer need to look for reports of grades, accesses and activities on several screens, since that a dashboard like the one proposed here is able to quickly inform those involved about the current performance of the selected sample.

As a way of continuing the work developed based on this, it is suggested that further research be carried out on the proposed model, implementing its dashboard in programming teaching institutions that have systems capable of retrieving information on indicators from educational databases.

---

## References

- 1 Liaqat Ali, Marek Hatala, Dragan Gasevic, and Jelena Jovanovic. A qualitative evaluation of evolution of a learning analytics tool. *Comput. Educ.*, 58(1):470–489, 2012. doi:10.1016/j.compedu.2011.08.030.
- 2 Jaqueline Vasconcelos Braga, Tiago Barros Pontes, Virginia Tiradentes Souto, et al. Statistical manipulations and visual anomalies: data visualization design and statistical bias recognition/manipulacoes estatisticas e anomalias visuais: design de visualizacao de dados e reconhecimento de vieses estatisticos. *Brazilian Journal of Information Design*, 17(2):145–163, 2020.
- 3 Alberto Cairo. *The truthful art: Data, charts, and maps for communication*. New Riders, 2016.
- 4 Doug Clow. An overview of learning analytics. *Teaching in Higher Education*, 18(6):683–695, 2013.
- 5 Abdullah Essa and Hany Ayad. Student success system: risk analytics and data visualization using ensembles of predictive models. In *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge*, pages 158–161. ACM, April 2012.
- 6 HU Hoppe. Computational methods for the analysis of learning and knowledge building communities. In *Handbook of Learning Analytics*, chapter 2. Society for Learning Analytics Research, 2017.
- 7 Nur Muizzatul Shafiqah Khuzairi, Zainal Choy Cob, and Tuan Hilaluddin. Towards understanding the synergetic relationship of data visualization with learning analytics: A review. In *AIP Conference Proceedings*, volume 2644(1), page 030030. AIP Publishing LLC, November 2022.
- 8 Simon Knight and Simon Buckingham Shum. Theory and learning analytics. In *Handbook of Learning Analytics*, pages 17–22. Society for Learning Analytics Research, 2017.

- 9 Vitomir Kovanović, Srećko Joksimović, Dragan Gašević, Marek Hatala, and George Siemens. Content analytics: The definition, scope, and an overview of published research. In *Handbook of Learning Analytics and Educational Data Mining*, pages 77–92. Society for Learning Analytics Research, 2017.
- 10 Lucas A B Macarini, Cristian Cechinel, Marilde F Batista Machado, Vanessa Faria Culmant Ramos, and Rodrigo Munoz. Predicting students success in blended learning—evaluating different interactions inside learning management systems. *Applied Sciences*, 9(24):5523, 2019.
- 11 Katerina Mangaroska and Michail Giannakos. Learning analytics for learning design: A systematic literature review of analytics-driven design to enhance learning. *IEEE Transactions on Learning Technologies*, 12(4):516–534, 2018.
- 12 Basilio Maraza-Quispe, Omar Marcelo Alejandro-Oviedo, Wilson Choquehuanca-Quispe, Nilton Cayturo-Silva, and Juan Herrera-Quispe. Towards a standardization of learning behavior indicators in virtual environments. *International Journal of Advanced Computer Science and Applications*, 11(11), 2020.
- 13 Isabel Meirelles. *Design for information: an introduction to the histories, theories, and best practices behind effective information visualizations*. Rockport publishers, 2013.
- 14 Rodrigo Paiva, II Bittencourt, Wagner Lemos, Andrade Vinicius, and Diego Dermeval. Visualizing learning analytics and educational data mining outputs. In *International Conference on Artificial Intelligence in Education*, pages 251–256. Springer, June 2018.
- 15 Rob Phillips, Dorit Maor, Greg Preston, and Wendi Cumming-Potvin. Exploring learning analytics as indicators of study behavior. In *EdMedia+ Innovate Learning*, pages 2861–2867. Association for the Advancement of Computing in Education (AACE), 2012.
- 16 Maren Scheffel, Hendrik Drachsler, Slavi Stoyanov, and Marcus Specht. Quality indicators for learning analytics. *Journal of Educational Technology and Society*, 17(4):117–132, 2014.
- 17 George Siemens. Learning analytics: The emergence of a discipline. *American Behavioral Scientist*, 57(10):1380–1400, 2013.
- 18 Felipe Cezar Cardoso Da Silva. Data visualization: past, present and future. *LIINC em Revista*, 15(2):205–223, 2019.
- 19 Dongmin Song. Learning analytics as an educational research approach. *International Journal of Multiple Research Approaches*, 10(1):102–111, 2018.
- 20 Vladimir L Uskov, Jon P Bakken, Karthik S Ganapathi, Kevin Gayke, Blake Galloway, and Johra Fatima. Data cleaning and data visualization systems for learning analytics. In *Smart Education and e-Learning 2020*, pages 183–197. Springer, 2020.



# Automated Assessment of Simple Web Applications

Luís Maia Costa  

Faculty of Engineering, University of Porto, Portugal  
CRACS – INESC TEC, Porto, Portugal

José Paulo Leal  

Faculty of Sciences, Universidade do Porto, Portugal  
CRACS – INESC TEC, Porto, Portugal

Ricardo Queirós  

uniMAD, ESMAD/P.Porto, Portugal  
CRACS – INESC TEC, Porto, Portugal

---

## Abstract

Web programming education is an important component of modern computer science curricula. Assessing students' web programming skills can be time-consuming and challenging for educators. This paper introduces Webpal, an automated assessment tool for web programming exercises in entry-level courses. Webpal evaluates web applications coded in HTML, CSS, and Javascript, and provides feedback to students. This tool integrates with Virtual Learning Environments (VLEs) through an API, allowing the creation, storage, and access to exercises while assessing student attempts based on the created exercises. The evaluation process comprises various subcomponents: static assessment, interface matching, functional testing, and feedback management. This approach aims to provide feedback that helps students overcome their challenges in web programming assignments. This paper also presents a demo showcasing the tool's features and functionality in a simulated VLE environment.

**2012 ACM Subject Classification** Applied computing → Interactive learning environments

**Keywords and phrases** Web Applications, Static Assessment, Interface Matching, Functional Assessment, Feedback Manager

**Digital Object Identifier** 10.4230/OASICS.ICPEC.2023.11

**Category** Short Paper

**Funding** José Paulo Leal is financed by National Funds through the Portuguese funding agency, FCT – within project LA/P/0063/2020. We also would like to acknowledge the European Union's Erasmus Plus programme (agreement no. 72020-1-ES01-KA226-VET-096004).

## 1 Introduction

The internet is a crucial resource for learning and researching nowadays, enabling teachers and students to access and share information. In the 1990s, Virtual Learning Environments (VLEs) were introduced to aid education, making it easier to share knowledge and provide feedback to students [10]. Evaluating students' programming abilities can be challenging since most exercises have an uncountable number of solutions. Moreover, since students must complete many programming tasks, it can burden teachers to assess them all [2]. This challenge increases in the automated assessment of web applications since these often use a combination of programming and markup languages, such as HTML, CSS, and JavaScript. Additionally, events in web applications do not follow a predetermined sequence and may occur randomly. As a result, conventional black-box methodologies that rely on test cases to assess student performance are inadequate for analyzing web interfaces and graphical user interfaces



© Luís Maia Costa, José Paulo Leal, and Ricardo Queirós;  
licensed under Creative Commons License CC-BY 4.0

4th International Computer Programming Education Conference (ICPEC 2023).

Editors: Ricardo Alexandre Peixoto de Queirós and Mário Paulo Teixeira Pinto; Article No. 11; pp. 11:1–11:8

OpenAccess Series in Informatics



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

## 11:2 Automated Assessment of Simple Web Apps

(GUIs) [2, 12]. To address these issues, this paper introduces a tool under development called Webpal (WEB Programming Assessment for Learning) that allows teachers to automatically assess basic web programming skills involving HTML, CSS, and JavaScript. While there are many tools and frameworks available to automate the assessment of various programming languages, there is a need for more solutions for web page evaluation. This work aims to develop an npm package that integrates with VLEs and assists in the automated assessment of web programming assignments and is capable of providing incremental feedback to the students about their attempts. The paper is structured as follows. Chapter 2 provides an overview of the current state of the art on tools and algorithms comparable to Webpal. Chapter 3 describes the Webpal architecture, features, and evaluation process. Lastly, Chapter 4 summarizes the significant contributions and highlights potential future directions for research in this area.

### 2 State of the Art

In this study, we introduce a tool for evaluating basic web applications, specifically involving HTML, CSS, and JavaScript. To the best of our knowledge, no equivalent tools exist that can offer an automated assessment within these distinct programming languages. As such, our analysis focuses on tools that embody attributes relevant to our project, particularly those relating to interface and functional assessments. It is noteworthy that most of the prevailing literature in the interface assessment domain seems to come from the early 2000s, with a predominant emphasis on tools developed in Java.

Douce, Livingstone, and Orwell [3] reviewed various automated assessment tools and pointed out some advantages and disadvantages. On the positive side, since assessing programming assignments is laborious, these tools can reduce teachers' work, allowing them to concentrate on more critical tasks, such as clarifying CS and programming concepts. Also, computers are less likely to fail if configured correctly, while humans are error-prone. On the other hand, there are many restrictions on what can be assessed automatically, and the feedback provided by these tools needs to be revised for educational reasons [11].

In 2010, Ihantola et al. [6] presented a literature review of the current systems for automatically assessing programming assignments from 2006 to 2010. The authors recommended that new automated assignment systems should explain clearly how the tool functions and that the security of these systems should get more consideration. They also suggested that automated assessment tools should be open-source to avoid scattering, emphasized the need to integrate automated assessment into virtual learning environments, and identified the assessment of web applications as an area for future research.

It's possible to identify two types of assessment for programming assignments:

- Dynamic Assessment assesses a program by executing it. It requires a secure sandbox environment that doesn't interfere with the host's computer integrity. It's possible to check the program functionality by comparing the result with test data sets or its efficiency by monitoring execution-related activities such as execution time and memory usage [2].
- Static Assessment, contrarily to dynamic, assesses programming code without executing it. Adequate software code doesn't just guarantee proper execution; other critical elements must be considered, such as coding style and code smells, to check if the code contains the correct syntax. It can also check for plagiarism in the source code [2, 1, 9, 16].

## 2.1 Interface Assessment

In 2006, Gray and Higgins [5] introduced an approach to grading GUIs by analyzing the hierarchical relationship among interface components. This method extracts data from graphical elements and uses dynamic class loading in object-oriented programming languages to retrieve information about the interface's object structure. Then, a file specifies the required actions and objects to test and grade assignments.

Štěpánek and Šimková [17] described three algorithms for comparing the internal structure of HTML trees to detect duplicate interfaces (phishing) and assist in education. The first algorithm counts the number of occurrences of each element in a tree. The second algorithm calculates a ratio representing the percentage of reference paths in the compared interface. The third algorithm identifies the largest possible subtree in both interfaces and calculates a ratio between them.

Thackston [15] noted in 2020 that only some tools are available to grade web assignments. The author proposed a method that uses XPath queries to automate the assessment of HTML files. Although XPath has the potential for automatic evaluation, the author reported limitations in flexibility, incomplete coverage, and grading difficulties when the assignment is partially correct.

Leal and Primo [12] proposed a web interface matching algorithm that uses element mapping to identify and compare elements from different interfaces. This method extracts original and derived properties from the Document Object Model (DOM) API, reflecting spatial relationships between elements. The authors suggested refining the algorithm and extending it to evaluate interfaces visually and functionally, along with creating an incremental feedback manager.

## 2.2 Functional Assessment

Sztipanovits et al. [14] proposed an application to assess the functionality of web graphical user interfaces (GUIs). This solution requires a list of URL hosting submissions with the interfaces to evaluate and an Excel document defining the inputs and test cases.

The js-assess online playground [7] is a tool that automatically assesses Javascript programming exercises. It is a serverless tool that runs inside a Javascript engine and combines an online editor with libraries to evaluate functionality, style, programming errors, and software metrics. js-assess can evaluate Javascript functionality, style, programming errors, and software metrics, being unable to assess interfaces and styling. Due to the lack of a server, feedback cannot be stored, making it useful only for self-studying. This tool has a collection of assignments available and is implemented in pure HTML and Javascript, facilitating the integration of js-assess into any web page. The authors highlighted the importance of researching how Javascript should be taught, the libraries that should be used, and where Javascript should be introduced to students.

Mirshokraie [8] proposed an automated technique to generate test cases for individual Javascript functions and DOM event sequences, which focused on developing efficient and practical tests for JS web applications. The approach consists of a three-step process that involves dynamically exploring the application to deduce a test model, generating test cases, and creating test assertions automatically using mutation testing. The technique uses specific mutation operators to identify common JS programming errors and analyze only the application's necessary code. The study results showed that the generated tests covered an average of only 68.4%, demonstrating that this method is not particularly interesting in generating tests for web applications.

## 11:4 Automated Assessment of Simple Web Apps

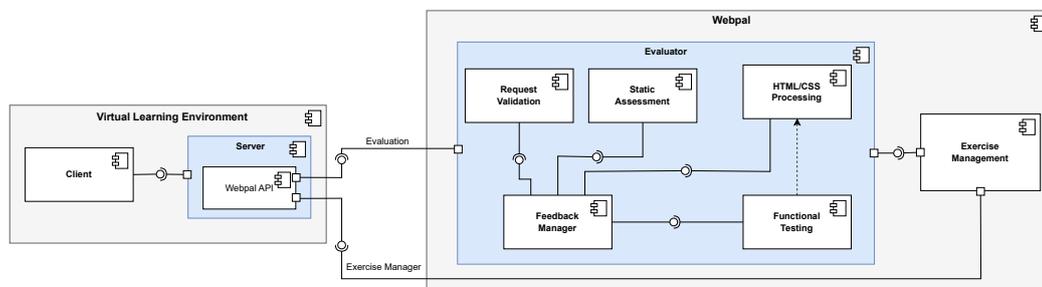
WebWolf [13] is a framework for automatically evaluating introductory web programming exercises. This tool can load web pages, find and inspect elements, click links, and perform unit testing. Teachers can create JUnit-like Java programs to test websites by writing test methods to simulate user inputs on the page and make assertions to compare the student's result with the expected output. However, WebWolf requires knowledge of Selenium's WebElement and the specification of IDs on web elements, which is not ideal. The authors concluded that this framework significantly reduces grading time compared to manual grading and has the potential for automated assessment.

### 2.3 Feedback Manager

There is a lack of information on systems incorporating incremental feedback managers. ProtoAPOGEE [4] is an automated grading system that aims to guide students and enhance faculty productivity by providing informative and iterative feedback through failed unit testing. The tool comprises three modules: project specification, automatic grading, and grading report viewer. The system requires teachers to know the Ruby programming language to write scripts. One challenge of providing detailed feedback to students was the need for teaching staff to write lengthy descriptive messages about errors. ProtoAPOGEE addresses this issue by generating an animation demo for each failed test case, providing a step-by-step guide on how the student failed a particular requirement. Teachers can also provide textual hints to assist students during the assignment. The tool can provide feedback at two levels: a summary report and detailed feedback for each requirement and test case.

## 3 Webpal

Webpal automates the evaluation process for web programming exercises in beginner-level courses. It assesses web applications coded in HTML, CSS, and Javascript, and provides immediate non-repetitive feedback to students. It consists of an independent module responsible for managing the entire tool, from file parsing to assessment and feedback generation. Webpal interacts with VLEs via an API, enabling the creation, storage, and access to exercises and assessing student attempts based on created exercises. The evaluation process encapsulates several subcomponents, including the analysis of the submission structure, code syntax verification, generation of a paired array of HTML elements from both the solution and attempted interfaces, and element mapping for functional testing. The latter employs chai/mocha tests provided during the exercise creation to evaluate the attempted web application. Each process can yield messages for the feedback generation manager, who processes these messages to provide the most helpful feedback for assisting students in overcoming their challenges.



■ Figure 1 Webpal architecture.

### 3.1 Webpal API

The Webpal API provides a collection of functions for managing and evaluating web development exercises. These functions include:

- `createExercise(exerciseData, testData, assignment)`: Returns a unique identifier after creating an exercise using the provided exercise data, test data, and assignment information.
- `getExerciseData(id)`, `getExerciseTestData(id)`, `getExerciseAssignment(id)`: Each function fetches specific information about an exercise using its unique identifier.
- `getAllExercises()`: Returns a list of all existing exercises.
- `deleteExercise(id)`: Deletes an exercise using its unique identifier.
- `evaluateAttempt(exerciseID, attemptData, PORT, previousFeedback)`: Evaluates an exercise attempt and returns the feedback.

The exercise and attempt data follow a structured JSON format where each submission consists of an array of objects. Each object includes two main properties: the filename and code. The filename is a string that represents the name of the file being submitted, while the code contains the actual content of the file, such as HTML, CSS, or JavaScript code. The test data, represented by a string of Mocha tests written in JavaScript, is used to evaluate the functionality of a learner's submission. Presently, Webpal assesses one HTML file per submission, including linked CSS and JavaScript files. Our future goal is to enable the evaluation of multiple pages per student submission. Webpal is deployed as an npm package, which simplifies its integration in existing JavaScript-based systems, and requires a server to execute.

### 3.2 Static Assessment

A preliminary step of Webpal `evaluateAttempt` function involves static evaluation. Three npm packages<sup>1</sup> are used for validating code syntax for each file type. For HTML, `html-validator` is employed, which validates HTML using Nu HTML Checker or `html-validate`. For CSS, `w3c-css-validator` is used, taking advantage of the W3C public CSS validator. Finally, `JSHINT` identifies Javascript errors and suggests better coding practices in JS. Static assessment helps to pinpoint issues before executing the evaluator, providing feedback about these problems without requiring Webpal to run the evaluator entirely, enriching students' understanding of their solution, and teaching them ways to produce more interoperable code.

### 3.3 Browser Emulation

Evaluating interfaces requires web page emulation to determine the relationships and positions of elements. Alternative approaches, such as `jsdom` and `cheerio`, were considered, which attempt to implement and emulate web standards on the server side. While these tools manage and verify relationships between HTML elements, they cannot simulate element positions. Webpal employs `Playwright`, a web testing and automation framework that runs headless browser instances, providing accurate results through the use of an actual web page. Another similar framework is `Puppeteer`, but it's older and lacks certain features, such as cross-browser support. The `express` npm package is used to serve files, offering a simple

---

<sup>1</sup> <https://www.npmjs.com/>

## 11:6 Automated Assessment of Simple Web Apps

solution for routing individual web pages and websites. Webpal examines the extensions of the specified files and express serves the HTML files locally on a port selected by the VLE. If CSS and JS files are imported into an HTML page, express will serve them. This dynamic routing approach enables the creation of various evaluators using different ports simultaneously.

### 3.4 Evaluation

The HTML/CSS processing component is responsible for processing attempt and solution HTML files, creating two tree structures representing HTML elements and associated CSS styles. This is achieved through a recursive traversal of elements and their children, ensuring all nested components are processed and included. The comparison and evaluation processes are simplified by utilizing a list of elements. Comparisons between elements and their properties are based on the Leal and Primo web interface matching algorithm [3]. This algorithm examines spatial relationships, styling, and textual content, resulting in a computed match score for each element pair within the attempt and solution trees. Pairs with the highest match scores are then selected. Subsequently, the functional testing component generates a mapping based on the best pairs, connecting the original IDs of the solution elements to their corresponding attempt element IDs. This mapping updates the functional test data provided initially (JS file with chai/mocha written tests), facilitating the automated testing of the attempt web page.

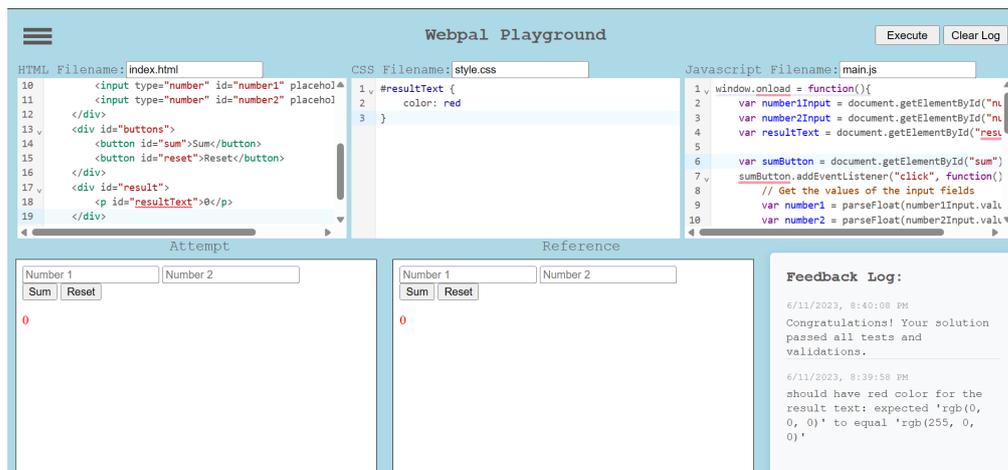
### 3.5 Feedback Manager

This module comprises a class that handles outputs sent by the request validation, static assessment, and functional testing modules. The Webpal execution follows a specific order, so it doesn't need complete execution if errors are encountered in the first steps of its pipeline. First, the student's attempt is checked against the predefined JSON schema to ensure it follows the required format. If the solution doesn't corroborate with the necessary structure, Webpal generates a feedback message highlighting this problem. Next, the static assessment module performs a syntax validation on the student's code. If any syntax errors are detected, a feedback message related to that error is generated. After the static evaluation and the HTML tree generation, the functional testing module tries to map the HTML elements of the student's attempt to the solution's HTML elements. The student is advised to review their HTML structure and element relationships if Webpal cannot find any correspondence between the two sets of elements. Finally, the functional testing module evaluates the functionality of the student's code, generating feedback for any failed tests.

### 3.6 Webpal Playground

The Webpal Playground is a demo created in Vue.js to showcase the features and functionality of Webpal, emulating a simplified Virtual Learning Environment. The demonstration consists of two pages: Playground and Backoffice. It is complemented by a backend server running on Node.js using the Webpal API. The Playground is the primary interface for students to interact with and explore Webpal. On this page, students can select from a list of available exercises and work on them using a built-in code editor. The code editor is divided into three sections for HTML, CSS, and Javascript, allowing students to edit and preview their work in real-time. It's also possible to check the solution output on this page. Students can execute and submit their code to be evaluated by clicking the "Execute" button. When

the code is submitted, Webpal evaluates the attempt and provides feedback to the student. The feedback is displayed in the “Feedback Log”, which records all feedback messages along with their timestamps. The Backoffice is designed for exercise management. Teachers can create new exercises on this page by providing an assignment name, exercise file names and code, and chai/mocha tests to evaluate the student’s code. Once an exercise is submitted, a success message is displayed, and the exercise is added to the list of available exercises in the Playground. The backend server utilizes the Webpal API to handle various tasks, such as creating, deleting, and fetching exercises, as well as evaluating student attempts. This server communicates with the frontend pages using HTTP requests and responses. Figure 2 illustrates the demonstration interface, presenting a scenario where two feedback messages are generated in response to two distinct attempts made by a student.



■ **Figure 2** Webpal playground demonstration.

## 4 Conclusion and Future Work

In this paper, we presented Webpal, an automated assessment tool for web programming exercises in introductory-level courses. Webpal focuses on evaluating student submissions coded in HTML, CSS, and Javascript, providing incremental feedback. The tool integrates with VLEs through an API. This work’s main contribution is an approach to assess the main aspects of simple web applications through static evaluation, interface matching, and functional testing. This approach also ensures that students receive valuable feedback to improve their programming skills and address the challenges they face in their assignments. Another contribution of this work was the development of an npm package that simplifies integration with VLEs and will serve as a base for validating this approach. There are several potential future directions for research in this field. It would be beneficial to extend the capabilities of Webpal to cover more advanced web programming assignments, such as multi-page web applications, or develop plugins to integrate directly with platforms like Moodle or Blackboard. The next steps include validating Webpal by integrating it into a code playground to evaluate the effectiveness of Webpal in terms of student learning outcomes and satisfaction to get insights of its impact. Comparing the performance of students using Webpal against traditional assessment methods could help establish the benefits of automated web programming assessment.

## References

- 1 Kirsti Ala-Mutka, Toni Uimonen, and Hannu-Matti Järvinen. Supporting students in C++ programming courses with automatic program style assessment. *JITE*, 3:245–262, January 2004. doi:10.28945/300.
- 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 Christopher Douce, David Livingstone, and James Orwell. Automatic test-based assessment of programming: A review. *J Educ Resour Comput*, 5, September 2005. doi:10.1145/1163405.1163409.
- 4 Xiang Fu, Boris Peltsverger, Kai Qian, Lixin Tao, and Jigang Liu. Apogee-automated project grading and instant feedback system for web based computing. *SIGCSE'08 – Proceedings of the 39th ACM Technical Symposium on Computer Science Education*, pages 77–81, 2008. doi:10.1145/1352135.1352163.
- 5 Geoffrey R. Gray and Colin A. Higgins. An introspective approach to marking graphical user interfaces. *ITiCSE06 – Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 2006:43–47, 2006. doi:10.1145/1140123.1140139.
- 6 Petri Ihantola, Tuukka Ahoniemi, Ville Karavirta, and Otto Seppälä. Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, Koli Calling '10, pages 86–93, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1930464.1930480.
- 7 Ville Karavirta and Petri Ihantola. Serverless automatic assessment of javascript exercises. *ITiCSE'10 – Proceedings of the 2010 ACM SIGCSE Annual Conference on Innovation and Technology in Computer Science Education*, page 303, 2010. doi:10.1145/1822090.1822179.
- 8 Shabnam Mirshokraie. Effective test generation and adequacy assessment for javascript-based web applications. *2014 International Symposium on Software Testing and Analysis, ISSTA 2014 – Proceedings*, pages 453–456, 2014. doi:10.1145/2610384.2631832.
- 9 Jan Nordin. A review on the static analysis approach in the automated programming assessment systems. *National Conference on Programming 07*, 2007. URL: <https://www.researchgate.net/publication/228328534>.
- 10 RL O'Leary and AJ Ramsden. *Virtual Learning Environments*, pages 1–30. Economics Learning and Teaching Support Network, 2002.
- 11 José Carlos Paiva, José Paulo Leal, and Álvaro Figueira. Automated assessment in computer science education: A state-of-the-art review. *ACM Transactions on Computing Education*, 22:1–40, September 2022. doi:10.1145/3513140.
- 12 Marco Primo and José Paulo Leal. Matching user interfaces to assess simple web applications. *OpenAccess Series in Informatics*, 91:7:1–7:0, 2021. doi:10.4230/OASIcs.ICPEC.2021.7.
- 13 Antonio C. Siochi and William R. Hardy. Webwolf: Towards a simple framework for automated assessment of webpage assignments in an introductory web programming class. *SIGCSE 2015 – Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 84–89, 2015. doi:10.1145/2676723.2677217.
- 14 Mate' Sztipanovits, Kai Qian, and Xiang Fu. The automated web application testing (awat) system. *Proceedings of the 46th Annual Southeast Regional Conference on XX, ACM-SE 46*, pages 88–93, 2008. doi:10.1145/1593105.1593128.
- 15 Russell Thackston. Exploring the use of xpath queries for automated assessment of student web development projects. *SIGITE 2020 – Proceedings of the 21st Annual Conference on Information Technology Education*, pages 255–259, 2020. doi:10.1145/3368308.3415389.
- 16 Nghi Truong, Paul Roe, and Peter Bancroft. Static analysis of students' java programs. *IFAC Symposium on Advances in Control Education*, pages 317–325, January 2004.
- 17 Jiří Štěpánek and Monika Šimková. Comparing web pages in terms of inner structure. *Procedia – Social and Behavioral Sciences*, 83:458–462, 2013. doi:10.1016/j.sbspro.2013.06.090.

# Haskelite: A Step-By-Step Interpreter for Teaching Functional Programming

Pedro Vasconcelos   

Departamento de Ciência de Computadores, Faculdade de Ciências da Universidade do Porto,  
Portugal

LIACC – Laboratório de Inteligência Artificial e Ciência de Computadores, Porto, Portugal

---

## Abstract

This paper describes *Haskelite*, a step-by-step interpreter for a small subset of Haskell. Haskelite is designed to help teach fundamental concepts of functional programming, namely: evaluation by rewriting; definition of functions using pattern-matching; recursion; higher-order functions; and on-demand evaluation. The interpreter is implemented in Elm and compiled to JavaScript, so it runs on the browser and requires no installation.

This is on-going work and has yet to be fully evaluated; we present some initial experience in the classroom and highlight some points for improvement.

**2012 ACM Subject Classification** Software and its engineering → Functional languages; Information systems → Web applications; Software and its engineering → Interpreters

**Keywords and phrases** Functional programming, Step-by-step evaluators, Web applications

**Digital Object Identifier** 10.4230/OASICS.ICPEC.2023.12

**Funding** This work was financially supported by: Base Funding - UIDB/00027/2020 of the Artificial Intelligence and Computer Science Laboratory - LIACC - funded by national funds through the FCT/MCTES (PIDDAC).

**Acknowledgements** The author would like to thank Sandra Alves for helpful feedback on a earlier draft of this paper.

## 1 Introduction

Functional programming has been taught in many universities as part of the Computer Science undergraduate curricula since the 1980s as attested by the steady tradition of textbooks available [2, 4, 9, 20, 1, 15]. Interest in functional programming is also increasing by the adoption of functional features in more mainstream languages, e.g. Java, Kotlin, Swift and Rust. Indeed, the ACM CS Curricula recommendation of August 2022 has increased the functional programming component from 3 to 4 hours relative to the CS2013 edition. However, educators have reported several beginner difficulties while learning functional programming, namely with the evaluation model [19], understanding type error messages [16, 21] or due to an incorrect perception of functional languages being only useful for academic problems [3, 14].

This paper describes *Haskelite*, a step-by-step interpreter for a small subset of Haskell [17]. Haskelite is designed to help with the first of the above problems, namely: understanding evaluation by rewriting; definition functions using pattern-matching and recursion; higher-order functions and on-demand evaluation. We focus on a restricted subset of Haskell covering these fundamental concepts. The Haskelite interpreter is implemented in Elm [6, 5] and compiled to JavaScript [10], so it runs on the browser and requires no installation. The source code is available at <https://github.com/pbv/haskelite> and a demo is available at <https://pbv.github.io/haskelite/>.

Haskelite is not intended to replace a full language implementation (so that students are not discouraged by learning only a “toy” language). It can nonetheless be useful for teachers and students to explain concepts and clarify misunderstandings.



© Pedro Vasconcelos;  
licensed under Creative Commons License CC-BY 4.0

4th International Computer Programming Education Conference (ICPEC 2023).

Editors: Ricardo Alexandre Peixoto de Queirós and Mário Paulo Teixeira Pinto; Article No. 12; pp. 12:1–12:8

OpenAccess Series in Informatics



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

# Summing squares

Summing the squares of numbers from 1 to 5. [Back](#)



```
sum (map square [1..5])
```

Evaluate

```
square x = x*x
```

line 1,col 23: expecting ), ,

■ **Figure 1** Haskelite interpreter embedded on a web page.

## 2 Design

### 2.1 Goals and motivation

The functional paradigm that we want to teach is well presented in the Bird and Wadler’s classic textbook [2]: “Programming in a functional language consists of building definitions and using the computer to evaluate expressions. The primary role of the programmer is to construct a function to solve a given problem. (...) *The primary role of the computer is to act as an evaluator or calculator.*” (emphasis added). In principle, this notion of computation by evaluation should be familiar to university students from high-school algebra; in practice, we have noted that students repeatedly experience difficulties:

1. There is a jump in complexity going from numerical expressions (as is done in high-school) to expressions with algebraic data types such as pairs and lists;
2. When defining functions by pattern matching, it is common for students to either miss necessary cases (e.g. the empty list) or to add redundant ones (e.g. special cases for lists of one or two elements);
3. Understanding recursive definitions requires transitioning from thinking by extension (e.g. viewing a list as a sequence of values [1,2,3,4]) to thinking inductively (e.g. viewing a list as the nested application of constructors  $1:(2:(3:(4:[])))$ );
4. Reasoning about termination also requires thinking inductively about “what is getting smaller” at each step of the recursion;
5. Reasoning about non-terminating processes (like lazy lists) also requires thinking (co-)inductively about transformations that consume or produce a few elements at a time.

Such difficulties can be overcome by diligently working out calculations on paper, but this often requires individual motivation and sometimes assistance by the teacher. The design goal for Haskelite is to help expedite this learning process by automating the calculation steps. Moreover, it should also allow the student to quickly change the expression and/or definitions and re-try the evaluation, hopefully improving his or her’s understanding.

Edit Reset < Prev Next >

```

sum (map square [1..5])
sum (map square [1,2,3,4,5])
sum ((square 1):(map square [2,3,4,5]))
(square 1)+(sum (map square [2,3,4,5]))
(1*1)+(sum (map square [2,3,4,5]))
1+(sum (map square [2,3,4,5]))
1+(sum ((square 2):(map square [3,4,5])))
1+((square 2)+(sum (map square [3,4,5])))
1+((2*2)+(sum (map square [3,4,5])))
1+(4+(sum (map square [3,4,5])))
1+(4+(sum ((square 3):(map square [4,5])))
1+(4+((square 3)+(sum (map square [4,5])))

```

sum (x:xs) = x+(sum xs)

■ Figure 2 Partial evaluation of the sum of squares.

## 2.2 User interface

Haskelite is a JavaScript application embedded in a web page. The user interface starts in *editing mode*, allowing students to type in equations defining functions and an expression. The editor provides feedback for parsing and type errors (see Figure 1). It is also possible for the instructor to prepare pages with filled-in expressions and definitions, so that the student is given particular examples to try.

Clicking the *evaluate* button switches to *evaluation mode*: the current expression is shown and the student can advance one step at a time; each evaluation step is either a primitive operation (e.g. arithmetic) or the application of a student function definition (or one from the *Prelude* of built-in functions). Previous evaluation steps are shown and a tool-tip highlights the *justification* for each step (see Figure 2). The student can also move backwards to previous steps.

Evaluation steps are justified as either primitive operations, one of the program's equations or an equation for a built-in *Prelude* function (such as `sum`). Using equations to justify evaluation steps serves two purposes: firstly, it connects the computation to the student's code; secondly, it helps prepare for the final topic of the course, which introduces proofs of program properties using equational reasoning and induction. This should help transition from *evaluations* with concrete values to *proofs* with variables that stand for arbitrary values.

## 2.3 Examples

We now present some examples of evaluations using Haskelite, highlighting possible uses in functional programming classes.

### 2.3.1 Structural recursion over lists

One of the early examples of recursion over lists is to (re)define the `sum` function that adds all values in a list:

```

sum [] = 0 -- base case
sum (x:xs) = x + sum xs -- recursive case

```

Haskelite can be used to exemplify evaluation of this function with some arbitrary lists. After this we can ask the students to define an analogous function to compute the product of all values a list. A frequent mistake is to define the base case incorrectly:

## 12:4 Haskelite: A Step-By-Step Interpreter for Teaching Functional Programming

```
product [] = 0 -- base case (WRONG)
product (x:xs) = x * product xs -- recursive case
```

Trying this in the Haskell interpreter gives only the final result (0); using Haskelite students can observe where the computation goes wrong:

```
product [1,2,3]
1*(product [2,3])
1*(2*(product [3]))
1*(2*(3*(product [])))
1*(2*(3*0))
1*(2*0)
1*0
0
```

The student's reaction might be to implement a less general solution where the base case is a list of a single value:

```
product [x] = x
product (x:xs) = x * product xs
```

This will, however, not work for the empty list; it is then possible for the teacher to compare with the sum function and relate the value of base case with the neutral element of operation in order to introduce the more general solution:

```
product [] = 1
product (x:xs) = x * product xs
```

### 2.3.2 Intercalating values

As a second example, consider the exercise: *Define a recursive function intersperse that intercalates a value between successive elements in a list.* For example: `intersperse 0 [1,2,3]` should be `[1,0,2,0,3]`.

This function requires a more complex recursion pattern because we need to distinguish whether the list has at least two values. The following attempt has a mistake in the recursive case:

```
intersperse a [] = []
intersperse a [x] = [x]
intersperse a (x:y:xs) = x:a:y:intersperse a xs
-- WRONG: should be x:a:intersperse a (y:xs)
```

Again evaluating the test case yields the following trace:

```
intersperse 0 [1,2,3]
1:(0:(2:(intersperse 0 [3])))
1:(0:(2:[3]))
1:(0:[2,3])
1:[0,2,3]
[1,0,2,3]
```

From this trace it should be easier to understand why the definition is wrong: each recursion step should remove only a single element from the list.

### 2.3.3 Mapping over an infinite list

As a final example, consider mapping the function that multiplies by 3 over the infinite list  $[1, 2, 3, \dots]$ . This illustrates both higher-order functions (`map`) and on-demand evaluation of infinite lists:

```
map (\x -> (3*x)) [1..]
map (\x -> (3*x)) (1:[2..])
((\x -> (3*x)) 1):(map (\x -> (3*x)) [2..])
(3*1):(map (\x -> (3*x)) [2..])
3:(map (\x -> (3*x)) [2..])
3:(map (\x -> (3*x)) (2:[3..]))
3:(((\x -> (3*x)) 2):(map (\x -> (3*x)) [3..]))
3:((3*2):(map (\x -> (3*x)) [3..]))
3:(6:(map (\x -> (3*x)) [3..]))
3:(6:(map (\x -> (3*x)) (3:[4..])))
3:(6:(((\x -> (3*x)) 3):(map (\x -> (3*x)) [4..])))
3:(6:((3*3):(map (\x -> (3*x)) [4..])))
3:(6:(9:(map (\x -> (3*x)) [4..])))
```

Evaluation will not terminate; instead the resulting list is also infinite but new elements are produced one at a time, so that the process is still algorithmic.

## 3 Implementation

### 3.1 Technical details

Haskelite was developed in the Elm programming language [6, 5], which compiles to JavaScript and runs in a web browser. No server-side software is required: the parsing, type-checking and evaluation all run in the web browser. Listing 1 illustrates the embedding of a Haskelite interpreter in an HTML page to evaluate the sum of squares example.

The techniques used are well-known: parsing is done using a parser combinator library<sup>1</sup>, type-checking implements the standard Hindley-Milner system [7] and evaluation is done by a naive rewriting engine using substitutions. The later choice is adequate: the focus is on the ability to easily relate the evaluation to the student's code (i.e. which equation is chosen) and efficiency is not critical for small programs and data sizes that a student is likely to use while learning.

The performance is perfectly adequate: the minified JavaScript bundle is only about 70KB of size, which is quite small by today's web page standards and parsing, type checking and evaluation of small programs appear instantaneous. Moreover, scalability to a large number of students is ensured because evaluation occurs in the clients' browsers.

■ **Listing 1** Embedding a Haskelite interpreter in an HTML page.

```
<!doctype html>
<html>
  <head><link rel="stylesheet" href="screen.css"></head>
  <body>
    <div id="haskelite"></div>
```

<sup>1</sup> <https://package.elm-lang.org/packages/elm/parser/latest/>

```

<script src="haskelite-min.js"></script>
<script>
  Elm.Haskelite.init({
    node: document.getElementById("haskelite"),
    flags: { expression:"sum (map square [1..5])",
            declarations: "square x = x*x" }
  });
</script>
<hr>
</body>
</html>

```

### 3.2 Limitations

Haskell is large language with a rich syntax and many extensions. Currently Haskellite supports only a very limited subset of Haskell:

- the only basic values are integers and booleans;
- the only structured data types are tuples and lists;
- definitions are by equations only (no case-expressions) and do not support *guards* (i.e. inline conditions);
- no support for user defined data types;
- no support for list comprehensions;
- no support for type classes.

Another important technical detail is that Haskellite implements a *call-by-name* evaluation strategy rather *call-by-need* (i.e. lazy evaluation), meaning that sub-expressions whose results would be shared in a lazy implementation will be re-computed in Haskellite. This is done to present each evaluation step as a simple expression; the alternative would be to expose an implementation using graph reduction, which would obfuscate the presentation. Due to the purely functional semantics, this change in evaluation strategy does not change the final result, only the number of evaluations steps required.

## 4 Related work

*Helium* is a special compiler and interpreter developed at the Utrecht University for teaching a subset of Haskell [13]. Its principal focus is on producing better messages for beginners, particularly type errors. This system appears to no longer be maintained<sup>2</sup>.

The Glasgow Haskell Compiler includes an interactive mode (GHCi) that can run programs using an imperative-style debugger, allowing setting break points, inspecting the values of variables and single-stepping execution [8]. It exposes Haskell's lazy evaluation mechanism, e.g. showing function arguments as partially evaluated expressions; thus it is intended more for experienced programmers than beginners.

*Python tutor* is a web site that allow visualizing the execution of Python, JavaScript, C, C++ and Java programs [12, 11]. The computational model is strictly imperative: the program state is visualized as pointer to the current instruction and the current values of variables in scope.

<sup>2</sup> As of April 2023, the *Hackage* database reports the last successful build in 2015: <https://hackage.haskell.org/package/helium>

There is a long tradition of teaching languages based on the functional programming language Scheme; *DrScheme* (now *DrRacket*) is an IDE for programming used for teaching which include a graphical debugger. This allows setting breakpoints, inspecting variables and step-by-step execution. There is even a web-based version [23]. However, Scheme does not encourage the reasoning by pattern-matching and equations that we are interested in teaching [22].

Haskelite was inspired by the *Lambda lessons* page built by Jan Paul Posma and Steve Krouse [18]. Lambda lessons allows step-by-step evaluation of simple definitions over lists and integers. The desire to extend this work lead to rewriting the interpreter from JavaScript into Elm.

## 5 Experience and further work

We started using Haskelite in the spring semester of 2022 and in only 2023 was the type-checker added. The interpreter has been used both in lectures and practical classes as soon as pattern matching and recursion are introduced.

The experience in practical classes is that students are keen to try out examples to clarify their understanding. Sometimes they will even try to use Haskelite as a debugger for larger programs (which will not work due to the limited language supported). We have not yet conducted any empirical validation of its application.

Nonetheless, there are already a number of directions for improvement that we have identified:

- adding more basic types to the language (e.g. strings and characters) would increase the range of examples that can be tried;
- similarly, adding support for user-defined datatypes would allow using it to explore more complex patterns of recursion, e.g. over tree-like data structures;
- the current user-interface side is not optimized for responsiveness in mobile devices; improving this should be relatively simple;
- there is currently no persistent state: expressions and definitions entered are lost each time the user closes the web page; adding the possibility of saving and loading programs using browser storage, or exporting links to share with others could be interesting in a classroom setting.

---

## References

- 1 Richard Bird. *Thinking Functionally with Haskell*. Cambridge University Press, 2015.
- 2 Richard Bird and Philip Wadler. *Introduction to Functional Programming*. Prentice-Hall, 1988.
- 3 Manuel M. T. Chakravarty and Gabriele Keller. The risks and benefits of teaching purely functional programming in first year. *Journal of Functional Programming*, 14(1):113–123, 2004. doi:10.1017/S0956796803004805.
- 4 Manuel M. T. Chakravarty and Gabrielle Keller. *An Introduction to Computing with Haskell*. Pearson SprintPrint, 2002. URL: <https://books.google.pt/books?id=qC3dAAAACAAJ>.
- 5 Evan Czaplicki. Elm: A delightful language for reliable web applications. <https://elm-lang.org/>, 2023. [Online; accessed June 2023].
- 6 Evan Czaplicki and Stephen Chong. Asynchronous functional reactive programming for GUIs. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 411–422, New York, NY, USA, June 2013. ACM Press.

- 7 Luis Damas and Robin Milner. Principal type-schemes for functional programs. In *Proceedings of the 9th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '82, pages 207–212, New York, NY, USA, 1982. Association for Computing Machinery. doi:10.1145/582153.582176.
- 8 GHC developers. The ghci debugger. [https://downloads.haskell.org/ghc/latest/docs/users\\_guide/ghci.html#the-ghci-debugger](https://downloads.haskell.org/ghc/latest/docs/users_guide/ghci.html#the-ghci-debugger), 2023. [Online; accessed June 2023].
- 9 Kees Doets and Jan van Eijck. *The Haskell road to Logic, Maths and Programming*. College Publications, 2004.
- 10 Mozilla Foundation. Javascript MDN. <https://developer.mozilla.org/en-US/docs/Web/JavaScript>, 2023. [Online; accessed June 2023].
- 11 Philip J. Guo. Python tutor. <https://pythontutor.com/>. [Online; accessed June 2023].
- 12 Philip J. Guo. Online Python tutor: Embeddable web-based program visualization for Cs education. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 579–584, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2445196.2445368.
- 13 Bastiaan Heeren, Daan Leijen, and Arjan van IJzendoorn. Helium, for learning Haskell. In *Proceedings of the ACM SIGPLAN Haskell Workshop (Haskell'03)*, Uppsala, Sweden, page 62. ACM SIGPLAN, August 2003. URL: <https://www.microsoft.com/en-us/research/publication/helium-for-learning-haskell/>.
- 14 John Hughes. Experiences from teaching functional programming at Chalmers. *SIGPLAN Notices*, 43(11):77–80, 2008. doi:10.1145/1480828.1480845.
- 15 Graham Hutton. *Programming in Haskell*. Cambridge University Press, 2nd edition, 2016.
- 16 Stef Joosten, Klaas Van Den Berg, and Gerrit Van Der Hoeven. Teaching functional programming to first-year students. *Journal of Functional Programming*, 3(1):49–65, 1993. doi:10.1017/S095679680000599.
- 17 Simon Marlow. Haskell 2010 language report. <https://www.haskell.org/onlinereport/haskell2010/>, 2023. [Online; accessed June 2023].
- 18 Jan Paul Posma and Steve Krouse. Lambda lessons. <https://stevekrouse.com/hs.js/>, 2014. [Online; accessed April 2023].
- 19 Judith Segal. Empirical studies of functional programming learners evaluating recursive functions. *Instructional Science*, 22:385–411, 1994.
- 20 Simon Thompson. *Haskell: The Craft of Functional Programming*. Addison-Wesley, 3rd edition, 2011.
- 21 Ville Tirronen, Samuel Uusi-Mäkelä, and Ville Isomöttönen. Understanding beginners' mistakes with Haskell. *Journal of Functional Programming*, 25:e11, 2015. doi:10.1017/S0956796815000179.
- 22 Philip Wadler. A critique of Abelson and Sussman or why calculating is better than scheming. *SIGPLAN Notices*, 22(3):83–94, 1987. doi:10.1145/24697.24706.
- 23 Danny Yoo, Emmanuel Schanzer, Shriram Krishnamurthi, and Kathi Fisler. WeScheme: The browser is your programming environment. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, ITiCSE '11, pages 163–167, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/1999747.1999795.

# A Systematic Review of Teacher-Facing Dashboards for Collaborative Learning Activities and Tools in Online Higher Education

Tiago Romão 

University of Trás-os-Montes e Alto Douro, Vila Real, Portugal  
Universidade Aberta, Lisbon, Portugal  
INESC TEC, Porto, Portugal

Pedro Pestana 

Universidade Aberta, Lisbon, Portugal  
CITAR, UCP, Porto, Portugal

Leonel Morgado  

LE@D, CIAC, & CEG, Universidade Alberta, Lisbon, Portugal  
INESC TEC, Porto, Portugal

---

## Abstract

Dashboard for online higher education support monitoring and evaluation of students' interactions, but mostly limited to interaction occurring within learning management systems. In this study, we sought to find which collaborative learning activities and tools in online higher education are included in teaching dashboards. By following Kitchenham's procedure for systematic reviews, 36 papers were identified according to this focus and analysed. The results identify dashboards supporting collaborative tools, both synchronous and asynchronous, along categories such as learning management systems, communication tools, social media, computer programming code management platforms, project management platforms, and collaborative writing tools. Dashboard support was also found for collaborative activities, grouped under four categories of forum discussion activities, three categories of communication activities and four categories of collaborative editing/sharing activities, though most of the analysed dashboards only provide support for no more than two or three collaborative tools. This represents a need for further research on how to develop dashboards that combine data from a more diverse set of collaborative activities and tools.

**2012 ACM Subject Classification** Applied computing → Collaborative learning; Applied computing → Distance learning; Applied computing → E-learning

**Keywords and phrases** Dashboards, collaborative tools, collaborative activities, online learning, e-learning, computer-supported collaborative learning

**Digital Object Identifier** 10.4230/OASICS.ICPEC.2023.13

**Funding** This work was supported by the TRIO project funded by the European Union's Erasmus+ KA220-ADU – Cooperation partnerships in adult education programme under grant agreement no. KA220-ADU-1B9975F8.

## 1 Introduction

Online learning has grown in public interest in the last few years, largely due to the COVID-19 pandemic [34]. A popular online pedagogical strategy is collaborative learning, where students interact with each other and in groups, typically seeking consensus on a learning topic [35].

Although monitoring students' online collaborative activities makes teaching more effective, as in face-to-face learning [45], and even though many collaborative tools have some kind of monitoring functionalities, most teachers and tutors are discouraged of doing so. The main reason for this is that such monitoring requires devoting substantial time and effort, as it is primarily a manual task [41].



© Tiago Romão, Pedro Pestana, and Leonel Morgado;  
licensed under Creative Commons License CC-BY 4.0

4th International Computer Programming Education Conference (ICPEC 2023).

Editors: Ricardo Alexandre Peixoto de Queirós and Mário Paulo Teixeira Pinto; Article No. 13; pp. 13:1–13:12

OpenAccess Series in Informatics



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

## 13:2 A Systematic Review of Teacher-Facing Dashboards

Learning dashboards support online teaching by automating the extraction and analysis of data on students' interactions within collaborative tools used in online learning. However, most collaborative activities dashboards are student-facing, to assist in students' self-regulation or help students' improve their own learning outcomes, unlike learning dashboards in general, which are mostly teacher-oriented [50].

To support teacher monitoring, and consequently support pedagogical orchestration, teacher-oriented dashboards for collaborative learning contexts need to consider students' interaction data, known as collaborative activities indicators [3]. These may include forum participation, sentiment analysis, document editing or audio-visual conferencing, among significant diversity of online collaborative learning activities [41]. Interaction data may be extracted both from synchronous or asynchronous tools, learning management systems (LMS), and as other pieces of software, such as communication or collaborative writing tools [1, 31]. In this study, we identify which collaborative tools and what collaborative activities are supported by indicators in current teacher-facing dashboards, in the context of online higher education.

To achieve this objective, we applied Kitchenham's procedure for systematic reviews to answer two research questions: RQ1: "Which collaborative tools are supported by online higher education teaching dashboards?" and RQ2: "What indicators about collaborative activities are provided in online higher education teaching dashboards?"

## 2 Background

### 2.1 Dashboards

A dashboard is "a visual display of the important information needed to achieve one or more goals, consolidated and arranged on a single screen" [54].

In educational contexts, these are typically known as Learning Dashboards, aggregating indicators about individual learners or groups, the learning processes, and the learning context, regardless of their target users being students (as with StepUp! [48] and STEMscopes [39]), teachers (with dashboards like GradeCraft [25], or GLASS [36]), program chairs (e.g. [19]; [57]), administrators (e.g. LADA dashboard [23]) or educational researchers (e.g. moocRP [50]). Learning dashboards aim to support decision-making by educational context participants, as well as support students' motivation and prevent dropout [33].

### 2.2 Collaborative Learning

Collaborative learning is defined as two or more people working together toward a shared learning goal [28], a context where "learners have the opportunity to converse with peers, present and defend ideas, exchange diverse beliefs, question other conceptual frameworks, and are actively engaged", including by editing or improving artefacts, either directly or indirectly [28].

Collaborative learning, within the context of online learning, involves collaborative tools, either synchronous or asynchronous. These may be communication tools, such as Discord (2015), Slack (2013), Skype (2003) or Zoom (2011), discussion tools like blogs or forums, collaborative writing tools such as Google Docs (2006), Microsoft Office 365 (2011) or Zoho (2005), social networking/media sharing tools including Facebook (2004), Twitter (2006), Youtube (2005) or Prezi (2009), source code management tools like Github (2008) or Gitlab (2011), project management tools such as Asana (2011) or Trello (2011), and learning management systems (LMS), e.g. Moodle (2002), Blackboard (1997) and Canvas (2011) [1, 31].

Collaborative learning activities are actions, tasks or exercises that enable collaborative learning, commonly grouped as synchronous, asynchronous or blended, and classified as discussion activities (with indicators such as sentiment analysis) involving forum participation, interaction, or rating, communication activities like instant messaging, audio-visual conferencing or emails and collaborative editing or development of activities such as document editing, programming code submission, file sharing or project management [3, 31].

Monitoring students online collaborative activities makes teaching more effective by checking their learning process, the access and review of learning resources or by keeping track of their contributions and their quality [45]. Often this is a manual task that involves substantial time and effort [41].

### 3 Research Methods

The goal of this systematic literature review is to identify which collaborative tools and activities are currently supported by indicators in online higher education teaching dashboards. The framework that we applied was Kitchenham's systematic reviewing procedures [32].

Kitchenham's procedures have three main phases: the planning of the review, where the need for the review is stated and the review protocol is developed; the research questions and strategy used to search primary studies, which includes the search query or terms, which databases or journals and conference proceedings to include, study selection criteria, study quality assessment and data extraction strategy; and conducting the review, where the search for the research studies is carried out, the studies quality assessment is performed and the data extraction and synthesis is executed.

In this systematic review our research questions are: RQ1: "Which collaborative tools are supported by online higher education teaching dashboards?" and RQ2: "What indicators about collaborative activities are provided in online higher education teaching dashboards?"

The Population of this systematic review, under Kitchenham's procedure for the planning stage, are online higher education teaching staff (professors, teaching assistants, etc.), the Intervention target are the learning dashboards, the Comparison criterium for inclusion is the dashboards' usage, and the Outcome items are dashboard indicators about students' collaborative activities and collaborative tools from which those indicators are collected. Finally, the Context is online higher education.

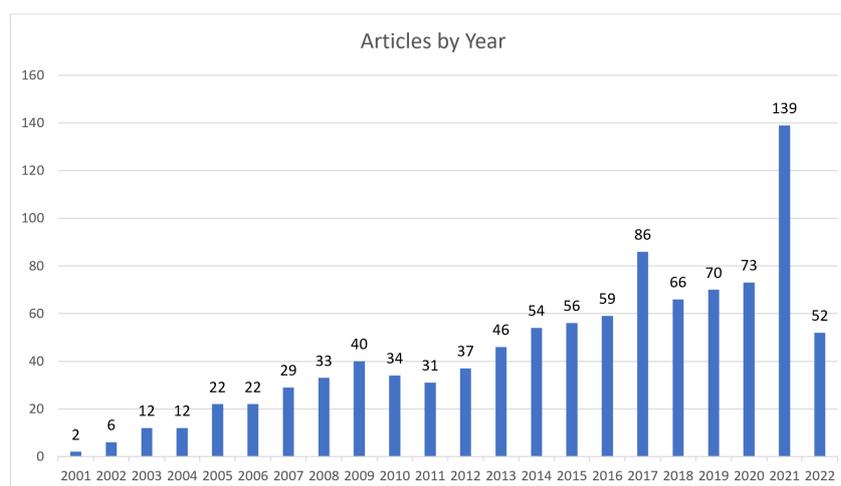
#### 3.1 Selection Process

The search query used was a combination of the keywords "online learning", "dashboard", "collaborative tools", "activities indicators" and their synonyms. The search process was conducted with the Publish or Perish software (available from <https://harzing.com/resources/publish-or-perish>) [24] using Google Scholar's database to look for the search terms in articles' titles and keywords.

We chose Google Scholar for this search process since nearly all citations found in other databases like Web of Science and Scopus were also found in Google Scholar's database and also that a substantial amount of unique citations were not found in these other databases [38]. However, since Google Scholar also includes less reputable sources, this required close scrutiny for reliability during the exclusion process.

### 3.2 Exclusion Process

In the exclusion process, duplicated articles were excluded. Then, because of the rapid evolution associated with social media, instant messaging and online learning collaborative tools, we considered only articles from the most recent 10 years, considering they represent 71% of all found articles, as shown in Figure 1, and some of the most used collaborative application were only launched recently, such as Discord in 2015, Slack in 2013, and Teams in 2017.



■ **Figure 1** Articles identified from the selection process by year of publication.

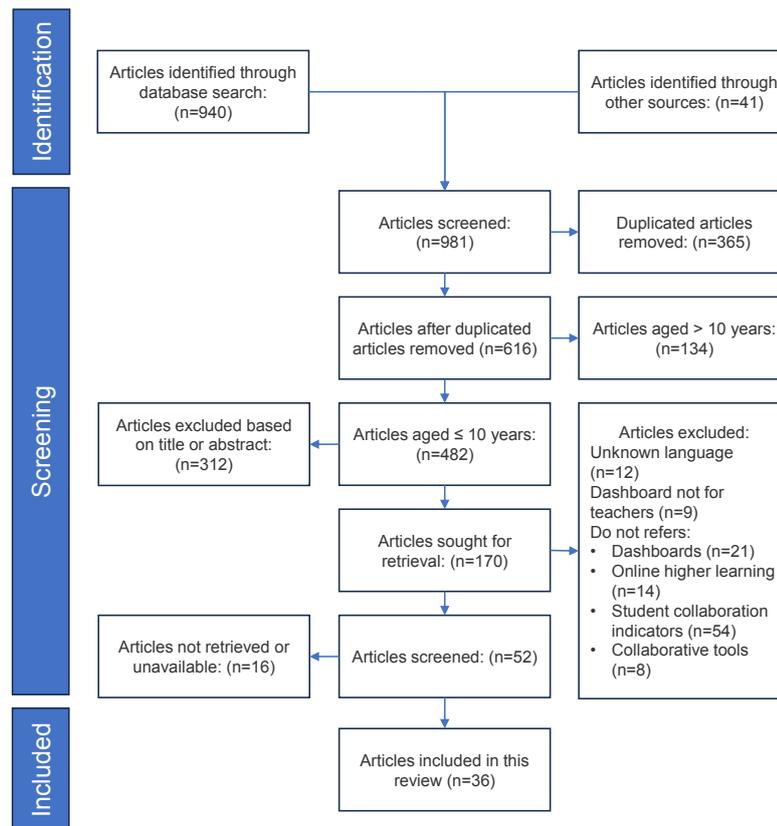
Afterwards, the titles and abstracts of the articles were analysed and those that were not relevant or did not make reference either to dashboards or to online higher education or to collaborative activities indicators were also excluded.

Finally, a more in-depth analysis was made by examining the entirety of the remaining articles. We excluded those not addressing teacher-facing dashboards (9 articles) or online higher education (14 articles), i.e. not meeting the the Population, Intervention or Context of this systematic review. We also excluded articles which did not mention collaborative platforms or indicators (62 articles).

As shown in Figure 2, a total of 981 articles were identified: 940 articles from the search queries and 41 from other serendipitous sources. From the selection process, 365 articles were found to be duplicates, 134 articles were more than 10 years old, 312 articles did not make reference either to dashboards, to online higher education or to collaborative activities indicators, based on the title or abstract. All these were excluded. There were 12 articles written in languages that the researchers are unable to read, i.e. neither in Portuguese, English, French, nor Spanish, and 122 articles that did not meet the selection criteria, resulting in 36 articles that were analysed for this systematic review (see Table 1).

## 4 Results and Discussion

As a result of the search process, both synchronous and asynchronous collaborative tools were used as analysis categories. These include communication tools, discussion tools, collaborative writing tools, social networking/media sharing tools, source code management tools, project management tools, and learning management systems [1, 31].



■ **Figure 2** Diagram of the systematic literary review.

We extracted elements from the final corpus of papers and subjected them to thematic content analysis, performing coding and theme development, leading to the following results.

Regarding RQ1, “Which collaborative tools are supported by online higher education teaching dashboards?” we found that the collaboration tools most used as data sources for online higher education teaching dashboards were learning management systems (and mainly Moodle), followed by social networking/media tools and collaborative writing tools (see Figure 3), appearing in 28 of the 36 articles.

The analysis categories utilised to achieve the second research question, RQ2: “What indicators about collaborative activities are provided in online higher education teaching dashboards?”, were collaborative learning activities such as discussion activities, communication activities and collaborative editing or development of activities [3, 31]. The discussion activities identified were forum participation, rating, and networking (mainly on LMS forums); the communication activities refer to synchronous communication, such as audio-visual conferencing, asynchronous communication (like email) or both synchronous and asynchronous communication using live chat or instant messaging by the use of tools like Slack or Discord.

The results regarding the collaborative activities’ indicators identified in the articles were mostly related to characterising LMS forum discussion activities, referenced in 31 articles of the total corpus, followed by indicators on live chat/instant messaging or other communication activities, and by indicators on collaborative editing or development of activities (see Figure 4).

## 13:6 A Systematic Review of Teacher-Facing Dashboards

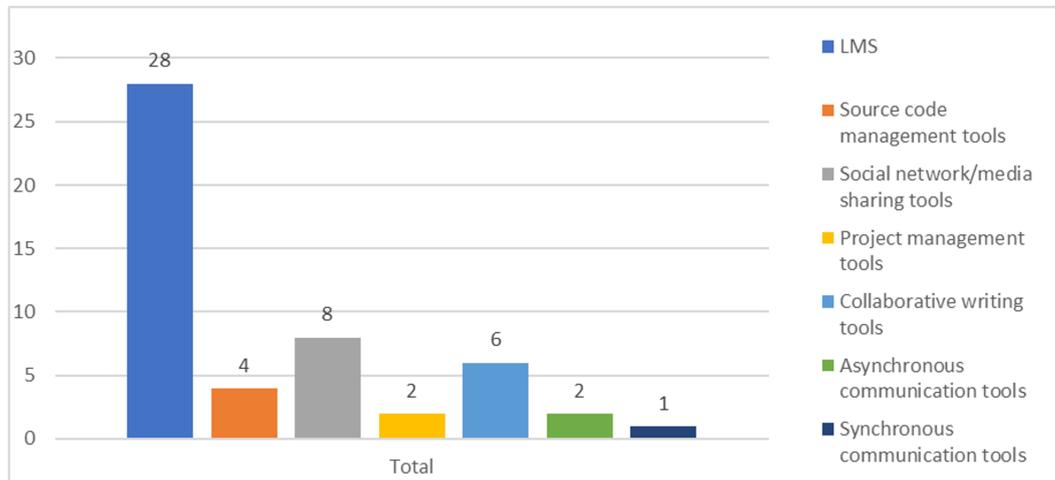
■ **Table 1** Overview of collaborative tools used in the dashboards from the reviewed articles.

Articles	LMS	Source code management tool	Social network/ media tools	Project management tools	Collaborative writing tool	Async. comm. tools	Sync. comm. tools
[2]	✓						
[4]	✓						
[5]	✓						
[6]	✓						
[7]	✓						
[8]	✓	✓	✓	✓	✓		
[9]	✓			✓			
[10]	✓						
[11]	✓		✓				✓
[12]	✓						
[13]	✓						
[15]	✓						
[16]	✓						
[17]	✓						
[18]		✓	✓	✓		✓	✓
[20]	✓						
[21]			✓				
[22]			✓				
[25]	✓						
[26]	✓						
[27]	✓						
[29]	✓						
[30]		✓					
[37]	✓						
[40]		✓					
[42]				✓		✓	✓
[43]		✓				✓	✓
[44]	✓	✓				✓	
[46]	✓	✓				✓	
[49]	✓	✓		✓	✓		
[51]	✓						
[14]		✓	✓				
[53]	✓						
[55]	✓						
[56]	✓						
[58]	✓						

Results on teaching dashboard indicators about collaborative activities in online higher education, were identified in the articles. These were mostly related to characterising LMS forum discussion activities, as referenced in 31 articles of the whole corpus of 36 articles. These provide relevant knowledge about student's interactions, namely by messages sentiment analysis and forum interaction networks. However, the asynchronous nature of forums does not provide the full dynamics of how students are communicating and collaborating with their peers. Particularly low in occurrences where indicators on project management, file sharing, submission of programming code, audio-video conferencing and email communication. Regarding this kind of collaborative learning indicators, we find there is a lack of research and possible future research is needed and encouraged. The least used collaborative tools as data sources in the teacher-facing learning dashboards were source code management tools, synchronous and asynchronous communication tools, and project management tools.

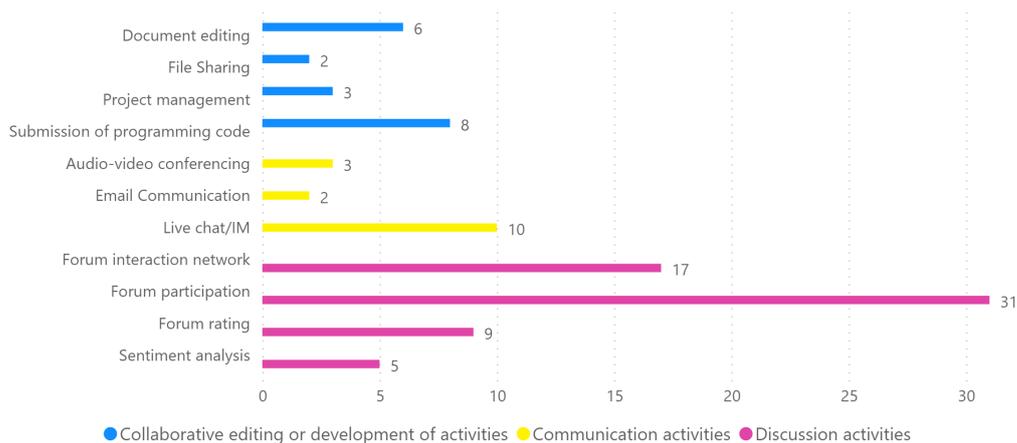
Regarding collaborative tools, most dashboards mainly used data from learning management systems exclusively and, to some extent, social network/media sharing tools or collaborative writing tools. By doing so, knowledge available from these dashboards is

somewhat restricted to the more strict or formal communication in the LMS and ignores other data sources, such as live chat/IM platforms or source code management tools. Thus, a more dynamic overview of the collaborating interaction between students is being ignored.



■ **Figure 3** Collaborative tools used in online higher education teacher-facing dashboards, by type.

In this systematic review, we identified a relationship between collaborative tools and collaborative activity indicators. The collaborative editing or development activities mostly take places in collaborative writing tools, project and source code managements tools, whereas discussion activities predominantly are carried out on the LMS. Regarding communications activities, email communication and live chat/IM, these take place in asynchronous communication tools, except for audio-video conferencing, which is synchronous in nature.



■ **Figure 4** Collaborative activities indicators identified in online higher education teaching dashboards.

## 5 Conclusions

In this study we identify which collaborative tools and activities are the source of indicators utilised in online higher education teaching dashboards.

The low occurrence of dashboard support for collaborative activities on asynchronous tools such as Slack or Discord is surprising, considering that none of the few indicators found in those categories used these collaboration tools as data sources, even though such collaboration tools are very popular among online higher education students, leveraging their social media-like features, and their combination of group and private messaging [47, 52]. A possible reason could be the pedagogical approach implemented in the institutions or a possible lack of knowledge or training on the part of the lecturers, something that was observed during the emergency online learning due to COVID-19 pandemic [45].

Additionally, several indicators on collaborative activities refer only to a single collaborative tool, notwithstanding that these indicators could be also implemented with data extracted from other data sources, in order to get a better understanding of student activity and interaction. An example of this is sentiment analysis, participation statistics, and interaction network indicators, which are mostly applied to LMS forum data, even though they could also use data from other collaborative tools, such as Slack or Discord, whereas live chat/IM communication activities indicators could also be applied to direct messages featured in most LMS.

We propose that future research efforts explore the challenges and opportunities of leveraging indicators about online collaborative learning activities taking place in the collaboration tools more frequently used in recent years, seeking to contribute to better understanding of collaborative learning in online contexts. This better understanding, in turn, is key to supporting better collaborative education and deploying collaborative learning.

---

## References

- 1 Abbas Abdoli-Sejzi, Baharuddin Aris, Maizah Hura Ahmad, and Mohd Shafie Rosli. The relationship between web 2.0 technologies and students achievement in virtual university. In *International Education Studies*, volume 8, pages 67–72, 2015. doi:10.5539/ies.v8n13p67.
- 2 Gokce Akcayir, Leticia Farias Wanderley, Carrie Demmans Epp, Jim Hewitt, and Athar Mahmoudi-Nejad. Learning Analytics Dashboard Use in Online Courses: Why and How Instructors Interpret Discussion Data. In *Visualizations and Dashboards for Learning Analytics*, pages 371–397. Springer, 2021. doi:10.1007/978-3-030-81222-5\_17.
- 3 Hosam Al-Samarraie and Noria Saeed. A systematic review of cloud computing tools for collaborative learning: Opportunities and challenges to the blended-learning environment. *Computers and Education*, 124:77–91, 2018. doi:10.1016/j.compedu.2018.05.016.
- 4 Naif Radi Aljohani, Ali Daud, Rabeeh Ayaz Abbasi, Jalal S. Alowibdi, Mohammad Basher, and Muhammad Ahtisham Aslam. An integrated framework for course adapted student learning analytics dashboard. *Computers in Human Behavior*, 92:679–690, 2019. doi:10.1016/j.chb.2018.03.035.
- 5 Elske Ammenwerth and Werner O. Hackl. Monitoring of students' interaction in online learning settings by structural network analysis and indicators. *Studies in Health Technology and Informatics*, 235:293–297, 2017. doi:10.3233/978-1-61499-753-5-293.
- 6 Haogang Bao, Yanyan Li, You Su, Shuang Xing, Nian Shing Chen, and Carolyn Penstein Rosé. The effects of a learning analytics dashboard on teachers' diagnosis and intervention in computer-supported collaborative learning. *Technology, Pedagogy and Education*, 30(2):287–303, 2021. doi:10.1080/1475939X.2021.1902383.

- 7 Francis Brouns, Marta Zorrilla, Elena Álvarez, and Pedro Solana-González. ECO D2.5 Learning analytics requirements and metrics report. Technical report, Open Universiteit, 2015. URL: [www.ecolearning.eu](http://www.ecolearning.eu).
- 8 Susan Bull, Matthew D. Johnson, Mohammad Alotaibi, Will Byrne, and Gabi Cierniak. Visualising multiple data sources in an independent open learner model. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7926 LNAI, pages 199–208. Springer, 2013. doi:10.1007/978-3-642-39112-5\_21.
- 9 Susan Bull, Michael Kickmeier-Rust, Ravi K. Vatrappu, Matthew D. Johnson, Klaus Hammermueller, William Byrne, Luis Hernandez-Munoz, Fabrizio Giorgini, and Gerhilde Meissl-Egghart. Learning, learning analytics, activity visualisation and open learner model: Confusing? In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8095 LNCS, pages 532–535. Springer, 2013. doi:10.1007/978-3-642-40814-4\_51.
- 10 Youness Chaabi and Yahya Al-Ashmoery. Development of a learning analytics extension in Open edX. In *International Conference of Modern Trends in ICT Industry: Towards the Excellence in the ICT Industries, MTICTI 2021*, pages 1–6. IEEE, 2021. doi:10.1109/MTICTI53925.2021.9664754.
- 11 Li Chen, Min Lu, Yoshiko Goda, Atsushi Shimada, and Masanori Yamada. Learning Analytics Dashboard Supporting Metacognition. In *Balancing the tension between digital technologies and learning sciences*, pages 129–149. Springer, 2021. doi:10.1007/978-3-030-65657-7\_8.
- 12 Miguel A. Conde, Francisco J. García-Peñalvo, Diego A. Gómez-Aguilar, and Roberto Theron. Visual learning analytics techniques applied in software engineering subjects. In *Proceedings - Frontiers in Education Conference, FIE*, volume 2015-February, pages 1–9. IEEE, 2015. doi:10.1109/FIE.2014.7044486.
- 13 Emilia Corina Corbu and Eduard Edelhauser. Responsive dashboard as a component of learning analytics system for evaluation in emergency remote teaching situations. *Sensors*, 21(23):7998, 2021. doi:10.3390/s21237998.
- 14 Rogério F. da Silva, Itana M. S. Gimenes, and José C. Maldonado. Analyzing learners' behavior and discourse within large online communities: a social learning analytics dashboard. In *Anais do XXXII Simpósio Brasileiro de Informática na Educação*, pages 576–585. SBC, 2021. doi:10.5753/sbie.2021.217468.
- 15 Jonce Dimov. Infrastructure of learning analytics dashboard. Technical report, Technische Universiteit Eindhoven, 2017.
- 16 Anna Dipace, Bojan Fazlagic, and Tommaso Minerva. The design of a learning analytics dashboard: EduOpen MOOC platform redefinition procedures. *Journal of E-Learning and Knowledge Society*, 15(3):29–47, 2019. doi:10.20368/1971-8829/1135044.
- 17 Raphael A. Dourado, Rodrigo Lins Rodrigues, Nivan Ferreira, Rafael Ferreira Mello, Alex Sandro Gomes, and Katrien Verbert. A teacher-facing learning analytics dashboard for process-oriented feedback in online learning. In *ACM International Conference Proceeding Series*, pages 482–489, 2021. doi:10.1145/3448139.3448187.
- 18 Xiaojing Duan, Chaoli Wang, and Guieswende Rouamba. Designing a learning analytics dashboard to provide students with actionable feedback and evaluating its impacts. In *International Conference on Computer Supported Education, CSEDU - Proceedings*, volume 2, pages 117–127, 2022. doi:10.5220/0011116400003182.
- 19 Anabela Costa e Silva, Leonel Morgado, and António Coelho. Drill-down dashboard for chairing of online master programs in engineering. In *Technology and Innovation in Learning, Teaching and Education - Second International Conference, TECH-EDU 2020, Vila Real, Portugal, December 2-4, 2020, Proceedings, 3*, pages 203–209, 2020. doi:10.1007/978-3-030-73988-1\\_15.
- 20 Luan Einhardt, Tatiana Aires Tavares, and Cristian Cechinel. Moodle analytics dashboard: A learning analytics tool to visualize users interactions in Moodle. In *Proceedings - 2016 11th*

## 13:10 A Systematic Review of Teacher-Facing Dashboards

- Latin American Conference on Learning Objects and Technology, LACLO 2016*, pages 1–6. IEEE, 2016. doi:10.1109/LACLO.2016.7751805.
- 21 Anatoliy Gruzd and Nadia Conroy. Designing a learning analytics dashboard for twitter-facilitated teaching. In *Proceedings of the 5th Annual ACM Conference on Learning at Scale, L at S 2018*, pages 1–4, 2018. doi:10.1145/3231644.3231704.
  - 22 Anatoliy Gruzd and Nadia Conroy. Learning analytics dashboard for teaching with Twitter. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, volume 2020-January, pages 2708–2717, 2020. doi:10.24251/hicss.2020.330.
  - 23 Francisco Gutiérrez, Karsten Seipp, Xavier Ochoa, Katherine Chiluiza, Tinne De Laet, and Katrien Verbert. LADA: A learning analytics dashboard for academic advising. *Computers in Human Behavior*, 107:105826, 2020. doi:10.1016/j.chb.2018.12.004.
  - 24 Anne-Wil Harzing. *The publish or perish book*. Tarma Software Research Pty Limited Melbourne, 2010.
  - 25 Caitlin Holman, Stephen Aguilar, and Barry Fishman. GradeCraft: What can we learn from a game-inspired learning management system? In *ACM International Conference Proceeding Series*, pages 260–264, 2013. doi:10.1145/2460296.2460350.
  - 26 Ya Han Hu, Chia Lun Lo, and Sheng Pao Shih. Developing early warning systems to predict students' online learning performance. *Computers in Human Behavior*, 36:469–478, 2014. doi:10.1016/j.chb.2014.04.002.
  - 27 Rangana Jayashanka, E. Hettiarachchi, and K. P. Hewagamage. Technology enhanced learning analytics dashboard in higher education. *Electronic Journal of e-Learning*, 20(2):151–170, 2022. doi:10.34190/ejel.20.2.2189.
  - 28 Heisawn Jeong and Cindy E. Hmelo-Silver. Seven affordances of computer-supported collaborative learning: How to support collaborative learning? How can technologies help? *Educational Psychologist*, 51(2):247–265, 2016. doi:10.1080/00461520.2016.1158654.
  - 29 Rogers Kaliisa and Jan Arild Dolonen. CADA: a teacher-facing learning analytics dashboard to foster teachers' awareness of students' participation and discourse patterns in online discussions. *Technology, Knowledge and Learning*, pages 1–22, 2022. doi:10.1007/s10758-022-09598-7.
  - 30 Max Kesselbacher, Kevin Wiltschnig, and Andreas Bollin. Block-based learning analytics repository and dashboard: Towards an interface between researcher and educator. In *ACM International Conference Proceeding Series*, pages 1–2, 2020. doi:10.1145/3421590.3421662.
  - 31 Hanan Khalil and Martin Ebner. Using electronic communication tools in online group activities to develop collaborative learning skills. *Universal Journal of Educational Research*, 5(4):529–536, 2017. doi:10.13189/ujer.2017.050401.
  - 32 Barbara Kitchenham. Procedures for performing systematic reviews. *Keele University*, 33(2004):1–26, 2004.
  - 33 Joris Klerkx, Katrien Verbert, and Erik Duval. Learning analytics dashboards. In Charles Lang, George Siemens, Alyssa Friend Wise, and Dragan Gašević, editors, *Handbook of Learning Analytics*, pages 143–150. Society for Learning Analytics Research, Alberta, Canada, 2017. doi:10.18608/hla17.012.
  - 34 Koushal Kumar and Bhagwati Prasad Pande. Rise of online teaching and learning processes during COVID-19 pandemic. In Lalit Mohan Goyal Praveen Kumar Khosla, Mamta Mittal, Dolly Sharma, editor, *Predictive and preventive measures for COVID-19 pandemic*, pages 251–271. Springer, 2021. doi:10.1007/978-981-33-4236-1\_14.
  - 35 Marjan Laal and Mozghan Laal. Collaborative learning: What is it? *Procedia-Social and Behavioral Sciences*, 31:491–495, 2012.
  - 36 Derick Leony, Abelardo Pardo, Luis de la Fuente Valentín, David Sánchez de Castro, and Carlos Delgado Kloos. GLASS: a learning analytics visualization tool. In *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge - LAK '12*, page 162, 2012. doi:10.1145/2330601.2330642.

- 37 SungTae Lim and EunHee Kim. The design of dashboard for instructor feedback support based on learning analytics. *The Journal of Korean Association of Computer Education*, 20(6):1–15, 2017.
- 38 Alberto Martín-Martín, Enrique Orduna-Malea, Mike Thelwall, and Emilio Delgado López-Cózar. Google Scholar, Web of Science, and Scopus: A systematic comparison of citations in 252 subject categories. *Journal of Informetrics*, 12(4):1160–1177, 2018. doi:10.1016/j.joi.2018.09.002.
- 39 Carlos Monroy, Virginia Snodgrass Rangel, and Reid Whitaker. STEMscopes: Contextualizing learning analytics in a K-12 science curriculum. In *ACM International Conference Proceeding Series*, pages 210–219, 2013. doi:10.1145/2460296.2460339.
- 40 Diana M. Naranjo, José R. Prieto, Germán Moltó, and Amanda Calatrava. A visual dashboard to track learning analytics for educational cloud computing. *Sensors (Switzerland)*, 19(13):2952, 2019. doi:10.3390/s19132952.
- 41 Donatella Persico, Francesca Pozzi, and Luigi Sarti. Monitoring collaborative activities in computer supported collaborative learning. *Distance Education*, 31(1):5–22, 2010. doi:10.1080/01587911003724603.
- 42 Stanislav Pozdniakov, Roberto Martinez-Maldonado, Shaveen Singh, Peter Chen, Dan Richardson, Tom Bartindale, Patrick Olivier, and Dragan Gasevic. Question-driven learning analytics: Designing a teacher dashboard for online breakout rooms. In *Proceedings - IEEE 21st International Conference on Advanced Learning Technologies, ICALT 2021*, pages 176–178. IEEE, 2021. doi:10.1109/ICALT52272.2021.00060.
- 43 Faisal A.T. Putra, Harry B. Santoso, and Rizal F. Aji. Evaluation of learning analytics metrics and dashboard in a software engineering project course. *Global Journal of Engineering Education*, 20(3):171–180, 2018.
- 44 A. Ramos-Soto, M. Lama, B. Vazquez-Barreiros, A. Bugarin, M. Mucientes, and S. Barro. Towards textual reporting in learning analytics dashboards. In *Proceedings - IEEE 15th International Conference on Advanced Learning Technologies: Advanced Technologies for Supporting Open Access to Formal and Informal Learning, ICALT 2015*, pages 260–264. IEEE, 2015. doi:10.1109/ICALT.2015.96.
- 45 Chrysi Rapanta, Luca Botturi, Peter Goodyear, Lourdes Guàrdia, and Marguerite Koole. Online university teaching during and after the COVID-19 crisis: Refocusing teacher presence and learning activity. *Postdigital Science and Education*, 2(3):923–945, 2020. doi:10.1007/s42438-020-00155-y.
- 46 Alex Rayón, Mariluz Guenaga, and Asier Núñez. Supporting competency-assessment through a learning analytics approach using enriched rubrics. In *ACM International Conference Proceeding Series*, pages 291–298, 2014. doi:10.1145/2669711.2669913.
- 47 Majid Rouhani. Utilizing Slack as a communication platform in a flexible learning trajectory course: Supporting the learning process. In *ACM International Conference Proceeding Series*, CSERC '20, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3442481.3442503.
- 48 Jose Luis Santos, Katrien Verbert, Sten Govaerts, and Erik Duval. Addressing learner issues with StepUp!: An evaluation. In *ACM International Conference Proceeding Series*, pages 14–22, 2013. doi:10.1145/2460296.2460301.
- 49 Jose Luis Santos, Katrien Verbert, Joris Klerkx, Sven Charleer, Erik Duval, and Stefaan Ternier. Tracking data in open learning environments. *Journal of Universal Computer Science*, 21(7):976–996, 2015.
- 50 Beat A. Schwendimann, Maria Jesus Rodriguez-Triana, Andrii Vozniuk, Luis P. Prieto, Mina Shirvani Boroujeni, Adrian Holzer, Denis Gillet, and Pierre Dillenbourg. Perceiving learning at a glance: A systematic literature review of learning dashboard research. *IEEE Transactions on Learning Technologies*, 10(1):30–41, 2017. doi:10.1109/TLT.2016.2599522.
- 51 Gabriel Lenon Barros Silva, Janaína Alexandre de Carvalho, and Alexandre Magno Andrade Maciel. Desenvolvimento de um learning analytics dashboard a partir de modelos de mineração

## 13:12 A Systematic Review of Teacher-Facing Dashboards

- de dados educacionais. *Revista de Engenharia e Pesquisa Aplicada*, 6(3):59–69, 2021. doi:10.25286/repa.v6i3.1688.
- 52 Keerthana Muthu Subash, Lakshmi Prasanna Kumar, Sri Lakshmi Vadlamani, Preetha Chatterjee, and Olga Baysal. DISCO: A dataset of Discord chat conversations for software engineering research. In *Proceedings - 2022 Mining Software Repositories Conference, MSR 2022*, pages 227–231, 2022. doi:10.1145/3524842.3528018.
- 53 Harmid Tarmazdi, Rebecca Vivian, Claudia Szabo, Katrina Falkner, and Nickolas Falkner. Using learning analytics to visualise computer science teamwork. In *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, volume 2015-June, pages 165–170, 2015. doi:10.1145/2729094.2742613.
- 54 Stephan D. Teasley. Student facing dashboards: One size fits all? *Technology, Knowledge and Learning*, 22(3):377–384, 2017. doi:10.1007/s10758-017-9314-3.
- 55 Saida Ulfa, Izzull Fattawi, Ence Surahman, and Hayashi Yusuke. Investigating learners' perception of learning analytics dashboard to improve learning interaction in online learning system. In *2019 5th International Conference on Education and Technology, ICET 2019*, pages 49–54. IEEE, 2019. doi:10.1109/ICET48172.2019.8987229.
- 56 Mehmet Uysal and Mehmet Barış Horzum. Designing and developing a learning analytics dashboard to support self-regulated learning. In *Visualizations and Dashboards for Learning Analytics*, pages 477–496. Springer, 2021. doi:10.1007/978-3-030-81222-5\_22.
- 57 Maria Viana, António Coelho, and Leonel Morgado. Drill-down and drill-through dashboard for master program chairing in e-learning. *Revista de Ciências da Computação*, 17:13–27, 2022. doi:10.34627/rcc.v17i0.279.
- 58 Juan Zheng, Lingyun Huang, Shan Li, Susanne P. Lajoie, Yuxin Chen, and Cindy E. Hmelo-Silver. Self-regulation and emotion matter: A case study of instructor interactions with a learning analytics dashboard. *Computers and Education*, 161:104061, 2021. doi:10.1016/j.compedu.2020.104061.

# An Experience with and Reflections on Live Coding with Active Learning

Anders Schlichtkrull  

Department of Computer Science, Aalborg University Copenhagen, Denmark

---

## Abstract

In this paper I report and reflect on a concrete experience with changing an introductory programming course from being based on “classical lectures” to being based on live coding with active learning. The experiment is built on learnings found in the literature and the pedagogical theories of scaffolding, think-pair-share and teaching as facilitation of learning. I reflect on the students’ reaction to the experiment, the difficulty of the active learning, how to keep time, coverage of learning objectives, the degree of improvisation and student involvement. The experiment was well received by the students, and I report also on the feedback. My hope is that educators who want to introduce live coding with active learning will be able to draw inspiration from my preparation of, execution of and reflections on the experiment.

**2012 ACM Subject Classification** Applied computing → Education

**Keywords and phrases** Live coding, active learning, teaching programming

**Digital Object Identifier** 10.4230/OASICS.ICPEEC.2023.14

**Acknowledgements** I thank my colleagues Kurt Nørmark and Jens Myrup Pedersen for providing me inspiration and ideas for changing the course and also for comments on an earlier version of this text. I thank my colleagues Johannes Bjerva, Søren Qvist Eliassen, Andrés Masegosa, David Maya-Drysdale and René Rydhof Hansen for discussions. I thank Aalborg University for giving me the opportunity to do this experiment as part of its University Pedagogical Program.

## 1 Introduction

In the fall of 2020, I taught the 5 ECTS course “Imperative Programming” (IMPR) with ~35 students at the first semester of the Software BSc education of Aalborg University Copenhagen (AAU CPH). For the lectures, a large seminar room was booked. In 2021 I would teach this course again – this time with ~55 students. The course material consists of a book [4], a set of video lectures for each week and a set of exercises for each week. The videos were prepared by my colleague Kurt Nørmark for his version of the course which runs on Aalborg University’s campus in Aalborg.

In 2020, each of my lectures in the seminar room consisted of some of the following parts:

- Classic lectures where I go through a number of slides to teach a curriculum.
- General feedback on a hand-in.
- Answers to questions from the videos.
- Live coding where the students see me write a program live in the seminar room.

I will focus here on the classical lecture and live coding. In my lectures, the “classic lecture” part took up a majority of the time. However, it frustrated me that I would often end up repeating the points that had already been taught in the video lectures. Some students also pointed this redundancy out to me in their feedback. On the other hand, I enjoyed particularly the live coding part because this gave the students a chance to see the concepts and theory that they had learnt about in the videos being applied to construct a program. Several students also told me that they learnt a lot from seeing a program being constructed.



© Anders Schlichtkrull;  
licensed under Creative Commons License CC-BY 4.0

4th International Computer Programming Education Conference (ICPEEC 2023).

Editors: Ricardo Alexandre Peixoto de Queirós and Mário Paulo Teixeira Pinto; Article No. 14; pp. 14:1–14:13

OpenAccess Series in Informatics



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

## 14:2 An Experience with and Reflections on Live Coding with Active Learning

One can also ask the question: What value can the lectures bring which videos cannot? With a traditional view of lectures where the active lecturer delivers knowledge to the passive students like a jug pouring water into a mug, the answer would be “Not much”. However, if we abandon this view, we see opportunities. When the students and teacher are in the same room it gives opportunities for active learning, immediate feedback and interaction.

With this in mind I decided to conduct an experiment in which my lectures would largely abandon the classical presentation of slides and instead expand the live coding. Additionally, I would try to have the students be active during the lecture and give them immediate feedback. One can discuss whether the word “lecture” can describe this approach, but for the lack of a better word I will keep using it for the remainder of this paper.

The rest of the paper is organized as follows: Section 2 states the problem considered. Section 3 presents the related work about live coding that the present paper builds on. Section 4 presents the concept of the experiment. Section 5 explains which pedagogical theories the experiment is based on. Section 6 explains how I prepared, presented, executed and continuously adapted the experiment. Section 7 reflects on the experiment. Section 8 presents my ideas for further experimentation. Section 9 discusses the feedback on the experiment that I got from the students. Lastly, Section 10 draws a conclusion on the experiment.

### 2 Pedagogical problem

Based on the above motivation I have formulated the following problem: “In the context of the course IMPR at AAU CPH, how can I teach programming using live coding and active learning?” I addressed the problem by experimentation in IMPR during the fall of 2021. In this paper I explain the experiment and reflect on how it went, and how I could go further with it.

### 3 Live coding

I am not the first to teach programming using live coding, and I draw inspiration for my experiment from many sources [5, 20, 15, 3, 12, 2], but in particular from Nederbragt, Harris, Hill and Wilson [9], Raj, Patel, Halverson and Halverson [13] and an example from the digital education platform Future Learn [7].

Nederbragt, Harris, Hill and Wilson [9] list ten quick tips for live coding. I list the tips here in *italic*, and explain the ones that are not self-explanatory:

1. *Go slowly.*
2. *Mirror your learner’s environment.* The setup that you use on the computer should be as similar as possible to that of the students.
3. *Be seen and heard.* You should not sit down behind a screen and hide. It is better to stand up and be seen and heard.
4. *Use the screen(s) wisely.* You should make sure that what you put on the screen can actually be seen, and therefore you should ensure that the font is big enough and the contrast strong enough e.g.
5. *Avoid distractions.* A main point here is to close programs that might give disturbing notifications during the lecture.
6. *Use illustrations – Even better, draw them.*
7. *Stick to the lesson material.*

8. *Embrace your mistakes.* When (not if) you make mistakes while live coding, you should explain the mistake to the students. In this way they will actually learn what to do when they end up in the same situation instead of just being confused.
9. *Get real-time feedback and provide immediate help.* The idea here is to let the students give real time feedback on whether they manage to follow along. Another point is to have a good ratio between students, teaching assistants and teachers.
10. *Turn learners into co-instructors.* Let the students be part of the process of writing the program by letting them give suggestions. You should also give the students a chance to discuss and put what they have learnt into words.

The authors also provide links to a video of a real world example of live coding [21], a (constructed) example of a teacher breaking most of the rules above [10], and an example of the same teacher following these rules [11].

Raj, Patel, Halverson and Halverson [13] explain the role of live coding in introductory programming. They point out a number of advantages:

- *Incremental coding.* Programmers do not sit and write a whole program before they run it. Rather they will change back and forth between writing the program and trying it out. Thus, the program is built incrementally. In slide presentations this will not be clear to the students because they will just see a perfect program “magically” appear on a slide.
- *Breaking down the problem.* Incremental coding can be done by breaking down the problem into smaller parts that can be solved individually. Live coding can illustrate this to the students.
- *Making thinking visible.* With live coding, the teacher has the chance to make thinking visible by expressing what thoughts go into building the program.
- *Modelling, Scaffolding and Fading.* Live coding presents an opportunity to do Modelling, Scaffolding and Fading. The authors see the modelling stage as the part where the teacher writes a program. They see the scaffolding stage as the part where the students have to solve an exercise based on what was learnt. They see the fading stage as when the students have to do home-work exercises without scaffolds.

The digital education platform Future Learn gives a small example of how one could do live coding to teach lists [7]. They give examples of how one can run the program often, ask questions to students, illustrate concepts and talk about errors in the program. I have also drawn inspiration from this example.

## 4 Concept

I present here my concept for the reorganized lectures of the fall of 2021:

- Introduction with slides
- 1st half
  - Live coding by Anders
  - Exercise in pairs for the students
  - Discussion of a solution
- 2nd half
  - Live coding by Anders
  - Exercise in pairs for the students
  - Discussion of a solution
- Small presentation with slides

The total time for this is ~2 hours. I gave the students 15 minutes to solve the exercises in pairs. After the lecture follows ~2 hours of group exercises in the same way as in 2020.

## **5 Pedagogical theory**

I will now explain the pedagogical theory behind the experiment. A recent survey by Rodrigues, Monteiro and Osório [14] identifies methodologies for teaching programming. Of these my experiment draws on active learning and peer programming, but also takes inspiration from the flipped classroom (via video lectures) and is situated in an educational program oriented towards Problem-Based Learning. It also relies on teaching in the lecture room – not as traditional classes with a slide deck, but rather by applying the learnings from the aforementioned papers on live coding and a number of other didactic tools explained below.

One can see teaching as facilitation of learning rather than mere delivery of knowledge. As written in the beginning, I see live coding with active learning as a way to do this. Of course, there is still knowledge that has to be delivered such as “What is a type?” and “How does an if-statement work?” but using live coding with active learning I explore a different way to facilitate the students’ learning of this knowledge than delivering it through a presentation.

Another pedagogical theory is that of scaffolding [22, 19], and as Raj, Patel, Halverson and Halverson [13] explained, live coding presents an opportunity to apply this principle. In the exercises in pairs the students have to program something that is similar to what they have just seen me program. Thus, the program I have just written can be seen as a scaffolding for the exercise. However, in contrast to Raj, Patel, Halverson and Halverson’s way to do live coding [13] my concept fades the scaffolds away in several stages rather than “just” in a set of home exercises. When the students arrive at the group exercises there will be less of a scaffold – these exercises are typically more challenging and are not preceded by a very similar example. There is still a scaffold however, namely the course material and the availability of the teacher and teaching assistant. Additionally due to the Aalborg PBL (Problem-Based Learning) model [16, 17, 18, 6, 1] used on the education program, the students will also use what they have learnt in their semester project. The 10 ECTS semester project module on the first semester consists of the students finding and delimiting a problem and programming a solution to it as a program in C. The project is done in groups of size approximately 7. Each group has a supervisor who is either a teacher employed by the university or hired externally. The problems that the students face in their semester projects are less structured than exercises, and the only scaffolding is the students’ supervision meetings.

A last pedagogical tool I will consider is “Think, pair, share” [8] in which lectures are broken up by pair exercises in which the students are given a question to think about, then pair up to talk about and lastly to share with the other participants. The methodology is generic and so I had to adapt it to the specific topic of live coding.

## **6 The experiment**

From the third week of the semester and forward, I conducted my experiment. For each week this consisted of preparing and executing that week’s live coding and exercises. In this section of the paper, I first explain my experiment as conducted in the third week and then I will explain how the experiment continued during the following weeks. In the following section, I will then give my reflections on the experiment.

## 6.1 Preparation

The third lecture's topic was logical expressions, if statements and switch statements. In my preparation I took a look at the learning objectives for that lecture and chose which ones of them I wanted to cover. I decided to cover logical expressions and if statements, but to leave out switch statements from the live coding. My reasoning was that it would simply be too much to also cover the switch statements and thus this part of the learning objectives would instead be covered "only" by the book, the videos and the exercises. Hereafter, I came up with 2 programs that illustrated respectively logical expressions and if statements. I then tried to write these programs as a practice. I also wrote a manuscript that contained the "plan" for writing the programs, i.e., how I would break it down into smaller parts and also suggestions for questions I could ask the students. Lastly, I made a minimalistic set of slides.

## 6.2 Presenting the experiment

In the "Introduction with slides" for this first run of the experiment, I presented and motivated the experiment to the students. I motivated live coding as a modern way to teach programming with the following advantages:

- The students see not only programs but also the process of constructing them.
- The students learn the thought process used to construct a program.
- The students see programs with mistakes and how these are repaired.

Furthermore, I argued that one of the learning objectives of the course is to program with the learnt knowledge, and that live coding could be a way to facilitate the students in achieving this objective.

After this presentation of the experiment, I asked the students if they were up for trying this and they agreed (by nodding and/or saying yes).

## 6.3 Execution

During the live coding I used large font sizes and also streamed the screen to the students' computers using Microsoft Teams. I did the streaming without sound, because the concept is that this is something we do in the seminar room. The motivation for streaming the screen was that everyone should be able to see what I code no matter where they sit in the room.

As shown in section 4, the concept of the lectures consists of two largely identical "halves" and therefore I will only explain the first of these. The first "Live coding by Anders" consisted of me programming a rock-paper-scissor game. During the development I asked the students questions and they also asked questions for me. The questions I asked were rather open, namely collecting suggestions on how we could split the program into smaller parts that we could then solve. Several students came up with suggestions.

In the first exercise in pairs, the students had to write an expression characterizing a draw in the game. During the 15 minutes where the students were working on this, I replied to their questions when they asked. After the exercise in pairs, I asked if anyone wanted to share a solution by uploading it to Microsoft Teams. Several pairs raised a hand, and I picked one. When they had uploaded it, I walked through it so that the rest of the students would also understand it. Then I pointed to a number of things that could be improved and showed how this could be done.

At the end of the lecture, I did an evaluation using an online form.

## **6.4 Continuing the experiment**

In the following weeks I continued and adapted the experiment. I made a number of changes such as:

- adjusting the difficulty of the exercises in pairs
- using the black board and slides for illustrations
- doing mistakes in the programming on purpose
- asking more closed questions compared to open questions.

The changes were based on reflections that I will discuss in the following section.

## **7 Reflection on the experiment**

I will now give my reflections on the experiment.

### **7.1 The students' reaction**

The students understood the motivation for the experiment and were ready to “play along”. Several students were also happy to give suggestions to my questions of how we could write the program. The students asked questions when there was something they did not understand, and they also remarked when they saw me make mistakes in my programming. During the exercises in pairs the students worked on them actively. The feedback I got from my survey was positive. It seemed that I had hit a good level of difficulty for the exercises in pairs, and feedback for the live coding was positive.

### **7.2 Bonus exercises in pairs**

When preparing the exercises in pairs, I had to consider the difficulty. Since the exercises have a length of 15 minutes, they have to be quite easy to be possible to finish within this time limit. I asked the students for feedback on the difficulty of the exercises. The feedback indicated that I managed to hit a spot where most student thought the difficulty level was just right, some thought it was too easy and some thought it was too hard. I also got personal feedback from some students that it was too easy for them. A challenge when teaching introductory programming is that students do not start with the same experience. There are students who meet up with no experience in programming, while others have experience from either hobby programming or from having computer science as a subject in high school. In order to also give these students a challenge I decided to introduce also bonus exercises. The idea was to include an optional objective for each of the exercises. My experience was, as hoped, that some students only did the normal exercises while others could also manage to do the bonus exercises. The only disadvantage I see with having bonus exercises was that when we discuss the solutions together, students would often submit solutions that solved also the bonus exercises, and perhaps this could confuse the ones who did only the normal ones.

### **7.3 Difficulty**

Another observation was of a subset of the students that had followed along with my live coding, completed the exercises in pairs during the lecture, but then at the group exercises really struggled to apply what they had been taught. At the group exercises I got the impression that they had managed to solve the exercises in pairs largely by copying what I had done and adapting it, but without really getting an understanding of what was going on.

I therefore wanted to change my exercises in pairs to avoid this problem. For this reason, I increased the difficulty of the future exercises in pairs. I made sure that each exercise had a small “twist” on what I presented in the live coding. E.g., in the lecture about structs, my live-coding example had structs where all values of fields were given as user input, whereas for the exercises in pairs, I required some of them to be calculated.

## 7.4 Keeping time

A challenge is to keep the time plan for each lecture. I set off 15 minutes for the students to finish the exercises in pairs and 5 for us discussing a solution. However, during the 15 minutes I would sometimes get a question in the last minute, and a dilemma was whether to help the student or to continue the lecture. If we were ahead of schedule and my impression was that most of the other students could use more time to work on the exercise I would just go ahead and help the student. In case we were not on schedule I would have to tell the student that I did not have time to help.

Another dilemma is that keeping the time can sometimes clash with Nederbragt, Harris, Hill and Wilson’s [9] tip of going slowly. Sometimes I simply did not have the time to go as slowly as I would have liked. Of course, I could have extended the lecture at expense of the group exercises, but this also has drawbacks.

## 7.5 Learning objectives

Related to the dilemmas of keeping time is the question of how many learning objectives to cover during a lecture. In the classical lectures with slides that I did in 2020 I could cover a lot! With slides I could introduce new concepts, example programs etc. at a high pace at the click of a button or the blink of an eye. However, if the students see many things at the blink of an eye how much do they really learn? Live coding forced me to cut down a lot on how many learning objectives I could cover in each lecture. The consequence was that I covered less, but I did it more thoroughly.

This was not without consequences though. As mentioned for the lecture on logical expressions, if statements and switch statements, I decided to not cover switch statements in the live coding. As a consequence, some students were confused in the lecture on loops, when I used a switch statement. Because of this experience, I decided to start every lecture by listing to the students which part of the week’s learning objectives were covered by the live coding, and which parts I expected them to learn from the videos, book and group exercises.

## 7.6 Sticking to the plan

Another challenge was when the students answered my open questions about how the program could be split into parts. The challenge was that they sometimes came up with a suggestion that was good, but that I had not prepared. Then I was in the dilemma of deciding between following what I had planned or the suggestion. I decided to compliment the student for their idea but to still follow my plan. Even if a student’s idea is good from a programming perspective, it might not illustrate certain parts of the language that the students have to learn. In these cases, my choice to carry on with my plan seemed to me to be the right one. In another case however, I decided to carry on where it might have been better to follow the student’s idea because in that particular case there would have been no loss with respect to what had to be learnt, and the student’s idea was actually better than mine.

## 7.7 Mistakes

Based on Nederbragt, Harris, Hill and Wilson's [9] tip of embracing mistakes, I did exactly that. Whenever there was a mistake, I would explain to the students what the mistake was and how we could fix it. The only downside of doing this is that it can take time to do, but I would say that it is worth it. I also experimented with doing mistakes on purpose. In a lecture on random numbers, I tried to make a program where I "forgot" to call the `rand` function on purpose. This meant that students would not only see a slide telling them to remember this function, but that they also saw in practice what happened if they forgot it. At the group exercises I was happy to see many students remember to call that function, but there were also students who still forgot it. I think this shows that for some students it is not enough to be told about the problem and to see someone else experience it. They need to experience it themselves. Therefore I would say that this shows the value of the group exercises.

## 7.8 Illustrating problematic programs

In 2020 I included in my lectures a part where I would, with consent, discuss during the lecture some, anonymous, program that the students had handed in. I tried to pick a program that was not perfect, and then I showed the students what could be improved. In the new concept I instead discussed the programs that the students did during the lecture. A problem here was that often only the students with a strong grip of the material would share their solution, and there would often not be much to say about these, since they were close to perfect. In the later lectures, however, some students were brave enough to volunteer with less perfect programs. These were more interesting to discuss because there was more to say about them. I will also point out that when I received these kinds of programs, I made sure to compliment them – in all cases the students were on the right track and I believe that the students should be rewarded for sharing a solution like this, no matter what state it is in.

## 7.9 Students as co-instructors

Nederbragt, Harris, Hill and Wilson's [9] give the tip of having students be co-instructors. In my lectures I saw this happening especially during the exercises in pairs. When the students work in pairs, students who understand the topic can help clear things up for those who don't, and additionally they get a chance to put their knowledge into words. From a logistical point of view, it also makes a lot of sense. In the lectures I am one teacher for ~50 students. In the group exercises we are one teacher and one teaching assistant for ~50 students. If I can facilitate the course such that students help each other, then the ratio is much better. I saw this happening during the exercises in pairs. However, there were also some students who preferred to work alone here, even though that was not the intention. In some of the lectures I tried talking to them about it during the exercises in pairs. Some then decided to find someone to work with, while others decided to keep working alone. Among those who decided to find someone to work with, I then saw them in the next lecture working alone again. But if this can help some of them to work in pairs then it is probably worth it.

## 7.10 Scalability

The following year, 2022, I taught the course again, but I was allowed to teach the course, in the same time slot, for also another educational program with ~40 students. Thus ~95 students started on the course that year. Because of this I was allowed to have two student

teaching assistants for the group exercises instead of only one. My experience was that the course scaled to the larger size without problems. In particular, the active learning part of the course gives all students the chance to be active during the lecture rather than only the ones who are brave enough to ask during the lecture or to answer one of my questions to the audience.

## **8 Possibilities for further experimentation**

I see a number of possibilities for further experimentation with teaching introductory programming using live coding and active learning. I will explain and reflect on them here.

### **8.1 Variations on the exercises in pairs**

In the exercises in pairs, I used a concept where the students work in pairs for 15 minutes. One can easily try out variations where they work for longer or shorter time in bigger or smaller groups. Another interesting idea could be to have the pairs meet other pairs to discuss their solutions. They could then look at pros and cons or to perhaps build a solution consisting of the best ideas of both solutions. The challenge with these opportunities is that one has to keep in mind both the time that they cost at expense of other activities and to which extent the facilities in the seminar rooms allow them to be possible. If the students sit on rows, then how can two pairs realistically meet up? In a seminar room that could perhaps work by having half of the pairs turn their chair, but in an auditorium it would not work.

### **8.2 Extending live coding with active learning**

In the 2021 iteration of the course, I did 2 hours of live coding with active learning followed by 2 hours of group exercises. An idea could be to extend the live coding to cover the full 4 hours. A challenge when doing live coding is that it takes time and I struggle to cover several learning objectives; more time could make it possible to cover more. I have also experienced that some few students leave after the live coding, and this could be a way to circumvent that problem. I do also see a number of negative aspects of this idea. Firstly, it removes the amount of time that is spent in the project groups and thus time where the group is bonded together. I think this would be a shame. Of course, one could have the students work in their groups during the lecture instead of the pairs, but unfortunately the facilities in the seminar rooms are not fit for that. Secondly, if we see the group exercises as a part of the course where the scaffolding is (partially) removed then one could also fear that abolishing this part would hinder the students' learning.

### **8.3 Going off script**

As discussed during the reflection, I largely avoided going off script during the live coding. I would like to explore what happens if I tried changing this habit. I suspect that the more I involve the students in the live coding, the more they will be active, and thus the more they will learn. One can also ask what the point is for me to ask them what we can do next if I am not willing to follow their suggestions.

### **8.4 Illustrating problematic programs**

As discussed during the reflection, it was more interesting to discuss a problematic program than a "perfect" one. Therefore I am wondering what opportunities there are for getting students to submit programs that are less than perfect. An idea could be to encourage it

## 14:10 An Experience with and Reflections on Live Coding with Active Learning

verbally during the lecture. Or a way could be to have all pairs submit a solution anonymously, and then I could look at them during a pause and pick out an interesting one to discuss. In this way it would not require bravery on behalf of the students to submit a solution.

### 9 Feedback from the students

In several of my lectures I asked the students for feedback on the lecture they had just experienced. I asked for quantitative feedback on the difficulty of the exercises in pairs and qualitative feedback on the lecture as a whole. In each of these I got replies from between 19 and 34 students. The general picture on the difficulty of the exercises in pairs was that I hit a good difficulty – there was a good spread between the students finding it easy, in between and difficult. In Appendix A, I list some of the concrete things that students liked about the live coding as well as some things they thought could be improved.

In addition, I also asked for feedback on the course as a whole in the last lecture. Here I got 33 replies. In the quantitative feedback, the students answered that they learnt a lot, were encouraged to participate actively, were motivated by the activities and liked the course. They also claimed to get a lot out of my (limited) presentation of slides, the live coding and the exercises during the lecture. All replies except 2 claimed to get a lot out of my feedback on a pair’s solution to an exercise during the lectures. In Appendix B, I list some of the concrete things that students liked about the live coding as well as some things they thought could be improved. In Appendix C, I have put bar charts for the quantitative feedback.

All in all, the feedback on this experiment from the students has been very positive – much, much more than I would have hoped for. When running the course again in the fall of 2022 for a larger audience the feedback was again very positive.

### 10 Conclusion

In the fall of 2021, I taught programming using live coding with active learning. I see this as an improvement over my previous approach where the emphasis was on teaching with slides. However live coding with active learning presented a number of challenges with respect to e.g. finding the right difficulty for the exercises, keeping time, coverage of learning objectives and illustrating problems in programs. The experiment was well received by the students. I see many opportunities for further experiments on the concept. It is my hope that teachers in programming can draw inspiration for their own teaching from my experiment, reflections and experiences.

---

#### References

- 1 Peter Dolog, Lone Leth Thomsen, and Bent Thomsen. Assessing problem-based learning in a software engineering curriculum using Bloom’s taxonomy and the IEEE software engineering body of knowledge. *ACM Transactions on Computing Education (TOCE)*, 16(3):1–41, 2016.
- 2 Alessio Gaspar and Sarah Langevin. Active learning in introductory programming courses through student-led “live coding” and test-driven pair programming. In *International Conference on Education and Information Systems, Technologies and Applications, Orlando, FL*, 2007.
- 3 Alessio Gaspar and Sarah Langevin. Restoring “coding with intention” in introductory programming courses. In *Proceedings of the 8th ACM SIGITE conference on Information technology education*, pages 91–98, 2007.
- 4 Jeri R. Hanly and Elliot B. Koffman. *Problem solving and program design in C (8th edition, global edition)*. Pearson, 2015.

- 5 Luke W Johnston, Madeleine Bonsma-Fisher, Joel Ostblom, Ahmed R Hasan, James S Santangelo, Lindsay Coome, Lina Tran, Elliott Sales de Andrade, and Sara Mahallati. A graduate student-led participatory live-coding quantitative methods course in R: Experiences on initiating, developing, and teaching. *Journal of Open Source Education*, 2(16):49, 2019.
- 6 Anette Kolmos, Flemming K Fink, and Lone Krogh. *The Aalborg PBL model: progress, diversity and challenges*. Aalborg University Press Aalborg, 2004.
- 7 Future Learn. Live coding. <https://www.futurelearn.com/info/courses/secondary-programming-pedagogy/0/steps/68423>, 2020. Accessed: 5. December 2021.
- 8 Frank T. Lyman. The responsive classroom discussion: The inclusion of all students. *Mainstreaming digest*, 109:113, 1981.
- 9 Alexander Nederbragt, Rayna Michelle Harris, Alison Presmanes Hill, and Greg Wilson. Ten quick tips for teaching with participatory live coding. *PLOS Computational Biology*, 16(9):1–7, September 2020. doi:10.1371/journal.pcbi.1008090.
- 10 Lex Nederbragt. A video introduction to live coding part 1. <https://www.youtube.com/watch?v=bXxBeNkKmJE>, May 2016. Accessed: 3. December 2021.
- 11 Lex Nederbragt. A video introduction to live coding part 2. [https://www.youtube.com/watch?v=SkPmwe\\_WjeY](https://www.youtube.com/watch?v=SkPmwe_WjeY), May 2016. Accessed: 3. December 2021.
- 12 John Paxton. Live programming as a lecture technique. *J. Comput. Sci. Coll.*, 18(2):51–56, December 2002.
- 13 Adalbert Gerald Soosai Raj, Jignesh M. Patel, Richard Halverson, and Erica Rosenfeld Halverson. Role of live-coding in learning introductory programming. In Mike Joy and Petri Ihantola, editors, *Proceedings of the 18th Koli Calling International Conference on Computing Education Research, Koli, Finland, November 22-25, 2018*, pages 13:1–13:8. ACM, 2018.
- 14 Gabryella Rodrigues, Ana Francisca Monteiro, and António Osório. Introductory programming in higher education: A systematic literature review. In Alberto Simões and João Carlos Silva, editors, *Third International Computer Programming Education Conference, ICPEC 2022, June 2-3, 2022, Polytechnic Institute of Cávado and Ave (IPCA), Barcelos, Portugal*, volume 102 of *OASICs*, pages 4:1–4:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/OASICs.ICPEC.2022.4.
- 15 Marc J Rubin. The effectiveness of live-coding to teach introductory programming. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 651–656, 2013.
- 16 John R. Savery. Overview of problem-based learning: Definitions and distinctions. *Essential readings in problem-based learning: Exploring and extending the legacy of Howard S. Barrows*, 9(2):5–15, 2015.
- 17 Maggi Savin-Baden. What are problem-based pedagogies? *Journal of Problem-Based Learning*, 2020.
- 18 Virginie Servant-Miklos. Problem-oriented project work and problem-based learning: “Mind the gap!”. *Interdisciplinary Journal of Problem-Based Learning*, 14(1), 2020.
- 19 Rob Wass and Clinton Golding. Sharpening a tool for teaching: the zone of proximal development. *Teaching in Higher Education*, 19(6):671–684, 2014.
- 20 Greg Wilson. Software Carpentry: lessons learned. *F1000Research*, 3, 2014.
- 21 Greg Wilson. Software Carpentry teaching demonstration. <https://vimeo.com/139316669>, 2015. Accessed: 3. December 2021.
- 22 David Wood, Jerome S. Bruner, and Gail Ross. The role of tutoring in problem solving. *Child Psychology & Psychiatry & Allied Disciplines*, 1976.

## 14:12 An Experience with and Reflections on Live Coding with Active Learning

### **A** Qualitative feedback from feedback form at the end of a number of lectures

Here is my summary of some concrete things that students mentioned they liked about the approach:

- Following along and understanding the live coding without prior coding experience.
- Live coding and active learning can be seen as “programming by practice”.
- Getting a chance to experiment with the code.
- Working in pairs gave a chance to learn from someone with more experience.
- The exercises in pairs as a way to be included in what is happening.

Here is my summary of some concrete suggestions for improvements:

- The live coding could be more detailed.
- One of the early lectures was quite easy.
- The tempo can be too high, which can encourage just copying what the lecturer was typing without understanding it.
- One could more often involve students in coming up with what to do next when live coding.
- Show an animation of what was happening *while* the lecturer was coding instead of doing it afterwards.

### **B** Qualitative feedback from feedback form at the end of the course

In the qualitative feedback, the students were positive about the live coding. Here is my summary of some concrete things that students liked about the approach:

- The live coding involves the students.
- The live coding shows the students how to build a program from scratch and not only the theory of programming.
- The live coding is engaging.
- The live coding made the content clearer and easier to implement.
- The live coding showed how to use the content in practice.

Here is my summary of the suggestions of what students thought could be better in the live coding:

- The lecturer could explain more of the thoughts that go into the programming.
- The tempo during the live coding can be too high for them to type along.
- The tempo during the live coding can sometimes be too high for students to understand what is going on.

**C** Quantitative feedback from feedback form at the end of the course

