LEARNING DESIGN WITH NEW CORE IDEA OF (TEACHING) PROGRAMMING

PR2 - REPORT

Co-funded by the
Erasmus+ Programme
of the European Union

Project	Object Oriented Programming for Fun
Project acronym	OOP4FUN
Agreement number	2021-1-SK01-KA220-SCH-00027903
Project coordinator	Žilinska univerzita v Žiline (Slovakia)
Project partners	Sveučilište u Zagrebu (Croatia) Srednja škola Ivanec (Croatia) Univerzita Pardubice (Czech Republic) Gymnazium Pardubice (Czech Republic) Obchodna akademia Povazska Bystrica (Slovakia) Hochschule fuer Technik und Wirtschaft Dresden (Germany) Gymnasium Dresden-Plauen (Germany) Univerzitet u Beogradu (Serbia) Gimnazija Ivanjica (Serbia)
Year of publication	2023

Table of contents

List of tables

List of figures

1. Introduction

The goal of this project result was to analyze and propose the innovative teaching and learning methods and approaches that could be used when teaching rather abstract concepts of object oriented programming in secondary schools. After we have identified the gap between the knowledge of finished secondary school students and the expectations from the university teachers, we would like to introduce a novel and contemporary approach in teaching object oriented principles in high schools.

Although it is rather challenging to define what is contemporary in the process of learning and teaching, since some old methods are becoming attractive, while some newer are replaced and modified, we suggest to start from the list presented in the The Book of Trends in Education 2.0, especially elements listed under chapter III Modern education is collaborative.

Several concepts are current buzzwords, such as project based learning, peer learning, team teaching, inquiry learning, flipped classroom, problem based learning, interdisciplinary learning, etc. Among the listed concepts, project based learning is one of the more comprehensive ones which, if applied in its full form, utilizes aspects which are present in other mentioned concepts. Following are several examples from each of the concepts:

- peer learning - during the project based learning students often work in teams, thus learning from each other. Teams also present their results so peer learning is further facilitated.
- team teaching - in project based learning teachers can work in pairs in the same classroom. This way they support interdisciplinary learning and facilitate holistic understanding of the topic at hand.
- inquiry learning - while working on their projects students often explore different questions which arise in the process. Inquiry learning can facilitate student's process of discovery, especially if they are seeking for new ideas or possible solutions.
- flipped classroom - goes hand in hand with demanding projects. Teachers can facilitate students in project development through guidance and instructions related to acquisition of basic knowledge at home, while more complex application and testing will be conducted in school with professional assistance from the teacher.
- problem based learning - projects are often related to real world problems and students need to identify the issue, deduct what they know and what they must solve (learn).
- interdisciplinary learning - projects which are used in project based learning are often complex and utilize knowledge from different subjects. This supports team teaching and coordination among teachers of different subjects. Further, such an approach provides support for the application of school knowledge in real world situations.

Also, a modern approach in designing lectures, especially those for elementary and highschool education is to define and to share *teaching scenarios.* Teaching scenarios (TS) "are perceived as a contemporary pedagogical approach which empowers individualisation of the teaching process by taking into consideration different student's needs. TS based teaching is focused on relevant knowledge and skills for the students, including those of need for the digital society. Careful planning of TS can remedy possible pitfalls and shortcomings which

might influence the teaching process." For better understanding of TS we recommend the following sources of relevant research and literature, such as *Hajdin, G. et al. (2022), Hajdin, G. et al. (2018)* and *Jerbić-Zorc, G. et al. (2021).*

In the context of this project result we would like to analyze best practices and prepare several teaching scenarios that would serve as an integral point of the following project activities including curriculum and teaching materials development in PR3, PR4 and PR5. Upon defining novel and innovative approaches and defining teaching scenarios we will initiate a *learning design* (LD) process by use of modern LD tools. This process will be finished in the later phases of the project after the curriculum is defined and will enable the team to have a comprehensive overview of the whole course along with the learning outcomes, topics, students' and teachers' activities and workload.

This chapter reports on the activities and results that we obtained during the work on second project result (PR2) and is designed as follows. Firstly, we describe unique teaching ideas by performing three comprehensive analyses aiming at: (1) analyzing teaching and learning materials, educational resources and existing teaching scenarios available in digital repositories and other sources; (2) performing a tertiary study on existing literature reviews in the domain of teaching and learning programming and (3) performing a semi-structured interviews with final-year students of Faculty of organization and informatics in order to understand what teaching and learning approaches they experienced during their study they find the most useful and results-giving. Additionally, we also give an overview of empirical results obtained from UNIZA who performed several workshops on using an innovative tool in teaching game programming and asked participants for feedback. Finally, we summarize the findings, align them with the results of gap analysis and propose novel teaching and learning scenarios in the form of teaching scenarios templates and learning design artifacts.

2. Methodology

zstapic@foi.hr

In this chapter, we outline the scientific methodology employed in our analysis, which aimed at unveiling unique teaching ideas through a multifaceted approach. The research design was carefully crafted to encompass four comprehensive analyses, each contributing a unique perspective to the analyzed area.

1. Analysis of Teaching and Learning Materials

To gain insights into existing teaching and learning practices, we initiated our research by meticulously analyzing a diverse array of teaching and learning materials. This included a thorough examination of digital repositories and other educational sources, allowing us to understand the prevailing pedagogical resources and methodologies in the domain of programming education. By scrutinizing these materials, we aimed to identify trends and innovative practices that could inform the development of novel teaching strategies.

2. Tertiary Study on Existing Literature Reviews

The second prong of our research methodology involved a comprehensive tertiary study of existing literature reviews within the domain of teaching and learning programming. This step was crucial for synthesizing the cumulative knowledge and insights provided by previous research efforts. By critically reviewing and analyzing relevant literature, we aimed to extract key findings, theoretical frameworks, and emerging trends. This not only helped us build a solid theoretical foundation for our analysis but also ensured that our analysis was aligned with the broader discourse in the field.

3. Semi-Structured Interviews with Final-Year Students

Recognizing the significance of firsthand experiences, we conducted semi-structured interviews with final-year students from the Faculty of Organization and Informatics. These interviews were designed to elicit detailed accounts of the students' experiences with various teaching and learning approaches throughout their academic journey. By engaging in open-ended discussions, we sought to understand the students' perspectives on the effectiveness and outcomes of different pedagogical methods. This qualitative approach allowed us to capture nuanced insights and uncover hidden aspects that may not be apparent through other analytical methods.

4. Analysis of Empirical Results from UNIZA

In addition to the aforementioned three analyses, our analysis involved the empirical investigation of the outcomes derived from workshops conducted at the University of Žilina (UNIZA). Three separate workshops were organized, each with participants from various high schools and different academic years. The primary objective of these workshops was to assess the motivational potential of a pedagogical approach integrating interesting topics, Object-Oriented Programming (OOP), and game development. The central question guiding this segment of the research was whether this combination could effectively generate interest in programming and STEM fields among participants.

3. Analyze Teaching/Learning Materials

dijana.plantak@foi.hr, mmatijevi@foi.hr, goran.hajdin@foi.hr, dperas@foi.hr, UNIZA, UNIBG

3.1. Introduction

In order to investigate teaching ideas in teaching programming that exist so far, the team has set a goal to explore educational resources (ERs) and existing teaching scenarios (TSs). In Croatia, there is a central repository of open educational resources with teaching scenarios for teachers and digital content for students in primary and secondary schools (https://edutorij.e-skole.hr/). However, for the field of informatics, there are only two such scenarios for the teachers in secondary school and one digital educational content for the students, so a proposition was made to explore educational resources in worldwide databases, such as Merlot.org (MERLOT) and OER Commons.

Digital repository Merlot.org is one of the oldest repositories with tens of thousands of discipline-specific learning materials for various educational levels, consisting of a wide variety of material types and technical formats. All the materials in MERLOT are reviewed for suitability for retention in the collection, some of them by editors and some of them by peer reviews.

OER Commons is a public digital library of open educational resources. Open Educational Resources (OER) are teaching and learning materials that can be freely used and reused at no cost, and without needing to ask permission. OER Commons includes over 50,000 high-quality OER, like full university courses, interactive mini-lessons and simulations, open textbooks but also lesson plans, worksheets and activities.

In the first phase of the exploration, the decision was made to review three educational resources in Croatian's Edutorij and explore MERLOT to gain insight about the structure and the content of existing educational resources indexed in that database, decide if deeper analysis of the particular material is needed and then to define additional criteria to perform deeper analysis of chosen materials. After reviewing MERLOT and Edutorij, the review of OER Commons resources was performed.

The next step was to perform additional analysis of chosen materials from all the repositories, to get better insight of used teaching methods and get possible directions and ideas for teaching approaches in teaching object-oriented programming. The final step will be inclusion of the partners in performing the analysis of their national repositories with the same set of criteria.

3.2. Conducting the analysis

When the MERLOT database was searched with the keyword "programming", 6,140 search results were shown. So, the search was narrowed with the following filters: audience (high school), review (editors), has no cost. Excluded from the search were commercial courses like Udemy and presentations from the conferences in video format. Another search with the same filters included keywords "game-based programming". Altogether, 43 links to the resources indexed in MERLOT were found.

All 43 materials were inspected and described in a table according to the following criteria: direct link to the learning material, years of upload and modification in MERLOT,

authorship, material type, technical/media format, short description of the material, identification of the content that could be useful for OOP4Fun (topic, innovative teaching, methods or tools) and a decision YES/NO if the material needs to be further explored. Materials included topics such as "C++ Tutorials", “PHP for the Novice Programmer”, "Greenfoot" etc. which can be seen in Figure 1. Preliminary analysis was partially done in Croatian language.

PR2 Tablica pregleda repozitorija Merlot.org
srpanj-kolovoz 2022.

Title of learning/teaching material with the URL on Merlot.org	Direct URL to the learning material	Date: added	Date: modified	Author(s)	Material Type	Media/Technical Format
Intro to Programming Concepts (using C)	http://faculty.valenciacollege.edu/carchibald/ComputerProgrammingConcepts.htm https://www.youtube.com/playlist?list=PL96BE5469318EFC74	2008	2021	Colin Archibald, Valencia College	Online course	video na YouTubeu
Alice Programming Language	http://www.alice.org/ http://www.alice.org/resources/	2005	2020	Stage 3 Research Group, Carnegie Mellon University	Kolekcija materijala	Web, aplikacija Alice2 i Alice3
C++ Programming Tutorial	https://www.thoughtco.com/candand-for-beginners-958278 https://www.thoughtco.com/c-and-c-plus-programming-4133470 (zastario)	2004	2019	John Kopp, About.com	Tekstualni članak	Web stranica

PR2 Tablica pregleda repozitorija Merlot.org
srpanj-kolovoz 2022.

Title of learning/teaching material with the URL on Merlot.org	Short description of the material	What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches (what kind of content, teaching method etc.)	Need for additional analysis?
Intro to Programming Concepts (using C)	The C language is used, along with the free programming tools from Microsoft - Visual Studio. This course is suitable for those who do not know any programming languages. It has no pre-requisites, and is suitable for college students, and high-school students who have completed high school algebra. Playlista s video sadržajima za predmet Intro to Computer .Programming Concepts. Struktura videa: voice-over ppt prezentacija + snimke ekrana iz korištenja Visual C++ 2005.	Vjerojatno ništa, video snimke su iz 2012. U playlisti nema naslova vezanog uz objektno-orijentirano programiranje.	NE
Alice Programming Language	Alice is a 3d graphics programming environment intended to be a gentle first introduction to students ranging from 6th grade to college… Using programming concepts and structures, students build 3D virtual worlds that are often compelling. In the Alice project, they acknowledge that capturing someone's attention is a pre-requisite to teaching them. Even a student's first programs can contain storytelling and game aspects, making the process of writing a program much more compelling, especially for female students.	Primjeri resursa: How to. Lessons, Excercises & Projects, Audio Library, Curriculum, Textbooks. Vrlo kvalitetni materijali. Vjerojatno ima i o o-o programiranju pa bi trebalo detaljnije pregledati. Izbacuje nekoliko materijala s ključnom riječi "object" - ako je to-to.	DA
C++ Programming Tutorial	This is a collection of 35 lessons for programmers beginning in C++. Lessons begin with the "Hello World" program and continue through classes.	Ništa se ne može iskoristiti jer poveznice ne sadržavaju isti sadržaj kao i ranije. Prva poveznica je jednostavno web stranica o jeziku C++, a druga nije originalni sadržaj.	NE

Figure 1. Preliminary analysis of educational resources in MERLOT about programming and game-base programming

Among 43 materials, there was also a material "Career Development Event Tabulations Program" and other 13 materials that had the word "program" in its title. After the preliminary analysis, those 14 resources were found to be irrelevant.

PR2 Tablica pregleda repozitorija Merlot.org srpanj-kolovoz 2022.			
Title of learning/teaching material with the URL on Merlot.org	**Short description of the material**	**What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches (what kind of content, teaching method etc.)**	**Need for additional analysis?**
Invasive Species | Chesapeake Bay Program	Whether introduced accidentally or on purpose, invasive species can cause harm to native plants and animals by encroaching on their food or habitat.The Chesapeake Bay ecosystem is seriously threatened by invasive species. There are more than 200 known or possible invasive species in the Chesapeake Bay watershed, competing with native plants and animals for food and habitat.	Nije programerski materijal.	NE
Spinoff Magazine: NASA's Technology Transfer Program	Spinoff highlights NASA technologies that benefit life on Earth in the form of commercial products.	Ništa. Radi se o dosta kratkim opisima primjera dobre prakse, bez konkretnog sadržaja.	NE
The leJOS Tutorial [program Lego robots with Java]	What is leJOS? Introduces the leJOS operating system and delivers some information on the environment your programs will run on. A Sample Application takes a closer look at a simple sample program; it introduces some general concepts of leJOS and the RCX. Common Problems (and their solutions) is the place to go when you are experiencing troubles compiling or running the sample. Exercises contains some exercises (and their solutions) where you can control the knowledge you gained from the precedent lessons.	Tutorijal o izradi aplikacija za upravljanje LEGO MINDSTORMS robotima. Sadrži popis naredbi s objašnjenjima. Nisam sigurna da se može iskoristiti. Zastarjeli materijali. Discusses classes of the leJOS platform that are essential for programming leJOS controlled robots.	NE
Media-Smart Youth: Eat, Think, and Be Active! Program Packet	"Media-Smart Youth: Eat, Think, and Be Active! is an interactive after-school education program that helps young people ages 11 to 13 understand the complex media world around them and how it can influence their health, especially in regard to nutrition and physical activity.	Nije programerski materijal.	NE

Figure 2. Identification of educational resources in MERLOT irrelevant for programming

A database OER Commons was searched with the same keywords as the database MERLOT: the first search with the keyword "programming" and the next search with a keyword "game-based programming". Both search results were narrowed with the filters similar to those used for filtering MERLOTs' results: audience (high school), user (teacher), subject area (computer science), material type (activity/lab, homework/assignment, interactive, lesson plan, module, teaching/learning strategy). Altogether, after removing duplicates **44** links to the resources indexed in OER Commons were found.

Resources were reviewed with the same criteria as in MERLOT (see Figure 3): direct link to the learning material, year of upload to the OER Commons, authorship, material type, technical/media format, short description of the material, identification of the content that could be useful for OOP4Fun (topic, innovative teaching, methods or tools), in addition to material license. Finally, a decision YES/NO was made if the material needs to be further explored.

PR2 Review of the repository EOR Commons September 2022						
Title of learning/teaching material with the URL on EOR Commons	**Direct URL to the learning material**	**Date added**	**License**	**Author(s)**	**Material Type**	**Media Format**
Climate Solutions Challenge	https://www.oercommons.org/courseware/lesson/75955/overview	26.12.2020	Creative Commons Attribution	Brian Henning	Activity/Lab, Lesson Plan	Downloadable docs
Simulating the Bug	https://www.oercommons.org/courses/simulating-the-bug/view	18.9.2014	Educational Use Permitted	Douglas Bertelsen	Activity/Lab	Text/HTML
Putting It All Together: Peripheral Vision	https://www.oercommons.org/courses/putting-it-all-together-peripheral-vision	18.9.2014	Educational Use Permitted	Anna Goncharova	Activity/Lab	Text/HTML
An Implementation of Steganography	https://www.oercommons.org/courses/an-implementation-of-steganography/view	18.9.2014	Educational Use Permitted	Brian Sandall, Derek Babb	Activity/Lab	Text/HTML
Let's code! Python Coding Examples :)	https://www.oercommons.org/courseware/lesson/68656/overview	17.6.2020	Creative Commons Attribution Non-Commercial Share Alike	Sertaç ATEŞ	Activity/Lab, Homework/Assignment, Unit of Study	Downloadable docs, Text/HTML, Other
Guide to PHP and MySQL	https://www.oercommons.org/courses/guide-to-php-and-mysql/view	28.1.2021	Creative Commons Attribution Non-Commercial Share Alike	Jim Gerland	Activity/Lab, Module, Reading, Textbook	Text/HTML
Exploring Acceleration with an Android	https://www.oercommons.org/courses/exploring-acceleration-with-an-android/view	18.9.2014	Educational Use Permitted	Brian Sandall, Scott Burns	Activity/Lab	Text/HTML

PR2 Review of the repository EOR Commons			
September 2022			
Title of learning/teaching material with the URL on EOR Commons	**Short description of the material**	**What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches (what kind of content, teaching method etc.)**	**Need for additional analysis?**
Climate Solutions Challenge	Project activity which focuses on students' familiarity with action items to reduce their carbon footprint.	Idea for project activity.	Yes
Simulating the Bug	Students modify a provided App Inventor code to design their own diseases. This serves as the evolution step in the software/systems design process.	It has a contemporary idea of disease transmission and additional resources, including application with code.	Yes
Putting It All Together: Peripheral Vision	In this culminating activity of the unit, students bring together everything they've learned in order to write the code to solve the Grand Challenge. The code solution takes two images captured by robots and combines them to create an image that can be focused at different distances, similar to the way that humans can focus either near or far. They write in a derivative of C++ called QT; all code is listed in this activity.	Code in C++, motivation and procedure of implementing it in the class.	Yes
An Implementation of Steganography	Students apply the design process to the problem of hiding a message in a digital image using steganographic methods, a PictureEdit Java class, and API (provided as an attachment). They identify the problems and limitations associated with this task, brainstorm solutions, select a solution, and implement it. Once their messages are hidden, classmates attempt to decipher them. Based on the outcome of the testing phase, students refine and improve their solutions.	Content related to steganographic methods.	No
Let's code! Python Coding Examples :)	Seven-problem set to use to practise Python Programming Language basics by solving problems.	Resources are in Turkish, although it is listed as English and Turkish.	No
Guide to PHP and MySQL	A guide for the PHP language and MySQL database for advanced web design course.	The text based guide has code examples for many elements like CSS, HTML/HTML5, JavaScript and PHP. The examples are generic.	Yes
Exploring Acceleration with an Android	Students conduct an experiment to study the acceleration of a mobile Android device. During the experiment, they run an application created with MIT's App Inventor that monitors linear acceleration in one-dimension.	The resource has android application developed for monitoring linear acceleration. Although the app is provided, most of the resources, including the assessment are actually focused on physics.	No

Figure 3. Preliminary analysis of educational resources in OER Commons about programming and game-base programming

In the third repository, Croatian repository Edutorij, three educational resources were found related to informatics topics in general: one was an educational resource aimed at the students and two were teaching scenarios for the teachers in secondary school. These materials were also reviewed according to the criteria set up for MERLOT and OER Commons (see Figure 4).

PR2 Review of the repository Edutorij					
September 2022					
Title of learning/teaching material with the URL on Edutorij	**Material Type**	**Media/Technical Format**	**Short description of the material**	**What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches (what kind of content, teaching method etc.)**	**Need for additional analysis?**
DOS, 1. razred SŠ - 3. Računalno razmišljanje i programiranje	digital educational resource	web site	Materijal za učenike za samostalno usvajanje sadržaja vezanih uz ishode učenja: definirati logički izraz za zadani problem; analizirati problem, definirati ulazne i izlazne vrijednosti te uočavati korake za rješavanje problema; primjenjivati jednostavne tipove podataka te argumentirati njihov odabir, primjenjivati različite vrste izraza, operacija, relacija i standardnih funkcija za modeliranje jednostavnoga problema u odabranome programskom jeziku. Materijal se sastoji od teksta, slika, videa i interaktivnih objekata.	All :-)	DA
SP, 1. razred SŠ - Ako budem imao sreće da ostvarim samo neke od svojih ideja, to će biti dobročinstvo za cijelo čovječanstvo Nikola Tesla, aktivnost - Volontiraj, mentoriraj, programiraj	teaching scenari	web page	Scenarij poučavanja koji na suvremen način opisuje moguću realizaciju ishoda učenja usklađenih s nacionalnim predmetnim kurikulumom.	All :-)	DA
SP, 1. razred SŠ, Razmijenimo iskustva u online okružju, aktivnost - Svaki tjedan zadatak jedan	teaching scenari	web page	Scenarij poučavanja koji na suvremen način opisuje moguću realizaciju ishoda učenja usklađenih s nacionalnim predmetnim kurikulumom.	All :-)	DA
				DA:	3

Figure 4. Preliminary analysis of educational resources in Edutorij with informatics topics

3.3. Presenting the results

In the first round of analysis, out of 29 learning/teaching materials related to programming or game-based programming in MERLOT and 44 materials in OER Commons, 10 materials were found relevant from MERLOT and 6 from OER Commons to be included in a deeper analysis.

The next step of the team was to explore those 16 materials + 3 from Edutorij with the focus on: **a)** topic of object-oriented programming and **b**) innovative pedagogical practices,

methods and tools such as problem based learning, inquiry-based learning, game-based learning, flipped classroom, project based learning, peer learning, etc… Materials were explored from the perspective of the lectures, exercises, tools and evaluation.

Each material was reviewed by two team members: one with the main background in programming and the other with the main background in didactics and/or e-learning methods. The goal was to identify good and innovative practices in teaching (object-oriented) programming that would contribute to PR2 goal: create learning designs with new teaching approaches in teaching of OO programming.

In the next tables, a brief description of nine (9) educational resources that were identified as valuable to contribute to the PR2 goal is given.

Alice Programming: Materials from Duke University (MERLOT)

Title of learning/teaching material with the URL on the repository	Alice Programming: Materials from Duke University
Direct URL to the learning material	https://www2.cs.duke.edu/csed/alice09/ Alice 2: https://www2.cs.duke.edu/csed/alice09/tutorials.php Alice 3: https://www2.cs.duke.edu/csed/alice09/tutorialsAlice3.php
Date: added	2015
Date: modified	2015
Author(s)	Susan Rodger, Duke University
Material Type	Online course
Media/Technical Format	Web, Alice 2 i 3 applications and tasks followed by video examples and Alice solution files.
Short description of the material	Alice is a tool that helps budding programmers to learn programming in a visual environment. These materials cover Alice 2 and Alice 3 and include videos, Powerpoint slides and handouts. Teachers can develop a short course in either of these Alice versions based on this material. Alice can also be used as a tool by teachers to develop lessons on topics such as reading, geography, math and other academic areas...
What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches	Alice offers a lot in the context of learning OOP. In particular, setting up a "computer game"/animation scene involves clarifying the context of the procedure and the class of objects, and visually shows the execution of the procedure on the objects. The form of the course can certainly be used. Thus, tasks with attached files of the beginning/end of solving and videos that deal with these tasks.
Need for additional analysis?	YES
Lectures	Materials cover approximately 15 hours of relevant OOP topics which are supported with clear structure, short video tutorials, slides, PDF-s and time estimates. This can be used either by teacher or student.
Exercises	Materials are not traditionally split between lectures and exercises but intervene those two concepts. Materials are practical, on-hand experience focusing on programming in Alice environment.

Tools	We suggest using Alice 3. Although Alice 2 is also shown in the materials, it's much older and not probably wouldn't be so appealing to younger pupils.
Evaluation	7 assessments are given in the form of starting scene, final scene and Powerpoint presentation for reaching the objective. No automatic grading is implemented, students are expected to manually check their solution.
Other important aspects	This material is very interesting because it connects video game development and OOP using simple coding blocks. It's easy to derive an actual object-oriented way of thinking using Alice and it's objects and scenes.
Final decision (what to use for OOP4FUN)	Yes, good structure and clear examples.

VB.Net teaching modules (MERLOT)

Title of learning/teaching material with the URL on the repository	VB.Net teaching modules
Direct URL to the learning material	http://web.archive.org/web/20071020032144/http://bpastudio.csudh.edu/fac/lpress/vbmodules/
Date: added	2005
Date: modified	2021
Author(s)	Larry Press, CSUDH (California State University Dominguez Hills)
Material Type	Collection
Media/Technical Format	Archived website with table of contents links
Short description of the material	This is a complete first course in programming, using VB.NET using VB.NET. Student outcomes using this material are slightly better than using a standard textbook.
What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches	The topic names/descriptions are very impressive, such as "Programming Terminology: Objects", "Object Terminology in the User Interface", "Object Terminology in the Real World", "Explaining Real Objects the Geek Way"... The meaning of the objects is excellently explained.
Need for additional analysis?	YES
Lectures	Lectures are structured in **topics**, mainly textual, but with visual examples when necessary.
Exercises	Named **Assignments**. There are practical examples of how objects are used and how they should be seen. From a programmer's perspective.
Tools	Named **Demonstration programs** related to the Assignments. They are simple Windows Forms applications that require no administration rights to run.

Evaluation	No classical questions or quizzes. Filling out the sentences is implemented in the Assignments.
Other important aspects	Interesting examples for each topic and exercise.
Final decision (what to use for OOP4FUN)	Yes, interesting examples for highschool students.

Greenfoot (MERLOT)

Title of learning/teaching material with the URL on the repository	Greenfoot
Direct URL to the learning material	https://www.greenfoot.org/door
Date: added	2008
Date: modified	2018
Author(s)	Poul Henriksen and Michael Kölling, University of Kent
Material Type	Simulation
Media/Technical Format	Web site
Short description of the material	Greenfoot teaches object orientation with Java. Create 'actors' which live in 'worlds' to build games, simulations, and other graphical programs. Greenfoot is visual and interactive. Visualization and interaction tools are built into the environment. The actors are programmed in standard textual Java code, providing a combination of programming experience in a traditional text-based language with visual execution.
What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches	Software for learning programming in a simple and fun way. There is a book: Introduction to Programming with Greenfoot, Object-Oriented Programming in Java with Games and Simulations. On the web is the Greenroom - a teacher community and provides resources (slides, worksheets, project ideas, tests, etc.) and a teacher discussion forum. You need to register and log it to look at the community because the OOP4Fun project application mentions Greenfoot.
Need for additional analysis?	YES
Lectures	Greenfoot does not have materials separated in traditional lectures and exercises. It does include relevant instructions which support the exercises.
Exercises	Exercises include motivation for students primarily focusing on games. They usually start with gaming elements in use and build upon the presented scenario or explain and extend its functionality.
Tools	Greenfoot is a comprehensive environment for development and run of games.
Evaluation	Not included. It could be easily created for project based learning.
Other important aspects	

Final decision (what to use for OOP4FUN)	YES

DOS, 1st grade high school - 3. Computer thinking and programming (Edutorij)

Title of learning/teaching material with the URL on the repository	DOS, 1. razred SŠ - 3. Računalno razmišljanje i programiranje
Direct URL to the learning material	https://edutorij.e-skole.hr/share/proxy/alfresco-noauth/edutorij/api/proxy-guest/e2fa002d-439a-484d-b2c6-6a3f4894a300/index.html
Date: added	2021.
Date: modified	2021.
Author(s)	Authors
Material Type	Digital educational resource (module)
Media/Technical Format	Web site
Short description of the material	Material for students to independently acquire content related to learning outcomes: define a logical expression for a given problem; analyze the problem, define input and output values and identify steps to solve the problem; apply simple data types and argue their selection, apply different types of expressions, operations, relations and standard functions for modeling a simple problem in the chosen programming language. The material consists of text, images, videos and interactive objects.
What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches	Everything :-)
Need for additional analysis?	YES
Lectures	Materials are prepared for stand-alone/independent/self learning. However, some of these materials could be used by the teachers during the lectures. materials Materials are not separated in traditional lectures and exercises. It does include relevant instructions which support the exercises.
Exercises	Exercises are mostly interactive and provide automated feedback to students. They include a variety of tasks from simple questions to simulation. Complex tasks require input from teaching.
Tools	Custom developed web based system which is limited on predefined set of modules for materials and exercised creation.
Evaluation	Materials include different assignments in both formative and summative aspects.
Other important aspects	
Final decision (what to use for OOP4FUN)	YES, although this might be quite different from what we will be doing.

Teaching scenario, 1st grade high school - If I'm lucky enough to realize just some of my ideas, it will be charity for all of humanity (Nikola Tesla), activity: Volunteer, mentor, program (Edutorij)

Title of learning/teaching material with the URL on the repository	SP, 1. razred SŠ - Ako budem imao sreće da ostvarim samo neke od svojih ideja, to će biti dobročinstvo za cijelo čovječanstvo Nikola Tesla, aktivnost: Volontiraj, mentoriraj, programiraj
Direct URL to the learning material	https://edutorij.e-skole.hr/share/proxy/alfresco-noauth/edutorij/api/proxy-guest/79b15720-defe-4487-965b-9010d63f06fb/index.html
Date: added	2021.
Date: modified	2021.
Author(s)	Authors
Material Type	Teaching scenario
Media/Technical Format	Web site
Short description of the material	A teaching scenario that describes in a modern way the possible realization of learning outcomes aligned with the national subject curriculum.
What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches	Everything.
Need for additional analysis?	YES
Lectures	There should be no traditional lectures. Teaching scenario focuses on students' motivation through relevant questions and real life situations. Presented elements combine learning outcomes and contemporary (hr. suvremene) pedagogical approaches.
Exercises	Teaching scenarios focus on student's activity through project based learning, problem based learning and other student-in-center approaches. Exercises can be related to just one section of a teaching session or to a longer term including several sessions, days or weeks... Exercises emphasize collaboration between students.
Tools	Teaching scenarios do recommend the set of tools that could be used. The tools are reviewed and evaluated and are appropriate for use in an educational environment. All those tools are free or part of functionality mentioned in teaching scenarios is free.
Evaluation	Evaluation is not strictly defined. Evaluation should be derived from other pedagogical aspects which are presented in the scenario. For example, if the teacher uses project based learning then evaluation should be adequate for that approach.
Other important aspects	
Final decision (what to use for OOP4FUN)	YES, the results from PR2 will be mostly similar to a novel teaching scenario for OOP4Fun.

Teaching scenario, 1st grade high school - Let's exchange experiences in the online environment, activity - One task every week (Edutorij)

Title of learning/teaching material with the URL on the repository	SP, 1. razred SŠ, Razmijenimo iskustva u online okružju, aktivnost - Svaki tjedan zadatak jedan
Direct URL to the learning material	https://edutorij.e-skole.hr/share/proxy/alfresco-noauth/edutorij/api/proxy-guest/a02d8fdc-dada-4eac-9dcb-cd9e86d080a5/index.html
Date: added	2021.
Date: modified	2021.
Author(s)	Authors
Material Type	Teaching scenario
Media/Technical Format	Web site
Short description of the material	A teaching scenario that describes in a modern way the possible realization of learning outcomes aligned with the national subject curriculum.
What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches	Everything.
Need for additional analysis?	YES
Lectures	Not related to OOP.
Exercises	Group work in a virtual classroom (Teams). For example, in our use-case, students could be divided into groups that represent classes. One student defines attributes for a specific class, another one a method, another student another method etc., and finally they're joined into groups by classes they defined.
Tools	Selection of IT tools used for collaborative learning (Teams, Wakelet) and evaluation (Google Forms).
Evaluation	No specific questions for evaluation.
Other important aspects	
Final decision (what to use for OOP4FUN)	Great example of using interesting innovative teaching methods.

Flow Charting App Inventor Tutorials (OER Commons)

Title of learning/teaching material with the URL on the repository	Flow Charting App Inventor Tutorials
Direct URL to the learning material	https://www.teachengineering.org/activities/view/uno_appinventor_lesson01_activity1
Date added	2014.
License	Educational Use Permitted
Author(s)	Brian Sandall, Rich Powers
Material Type	Activity/Lab
Media/Technical Format	Text/HTML
Short description of the material	Students design and create flow charts for the MIT App Inventor tutorials in this computer science activity about program analysis. Students work through tutorials, design and create flow charts about how the tutorials function, and present their findings to the class. In their final assessment, they create an additional flow chart for an advanced App Inventor tutorial.
What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches	This material introduces a software called "App Inventor" published by MIT. This software enables its user to create actual simple Android applications by only joining pieces of "coding puzzles". This software should be analyzed, but the material itself is of no use to us.
Need for additional analysis?	YES
Lectures	Short summary of **MIT App Inventor tool** with examples of designing and creating flow charts. There are Learning objectives, Material List, Prereq Knowledge, Introduction/Motivation, Procedure, Vocabulary/Definitions, Assessment.
Exercises	Creating simple apps for mobile devices without programming.
Tools	App Inventor: https://appinventor.mit.edu/
Evaluation	Detailed description of Assessment: Pre-Activity Assessment, Activity Embedded Assessment, Post-Activity Assessment.
Other important aspects	One good example is how they lay down all the components needed for UI in a simple table form: "Component type / Palette group / What you'll name it / Purpose". That way, students can define the entire UI by only reading the table.
Final decision (what to use for OOP4FUN)	Great example of using App Inventor (OOP) tool and detailed description of teaching methods.

Code.org (OER Commons)

Title of learning/teaching material with the URL on the repository	Code.org

Direct URL to the learning material	https://code.org/
Date added	2017.
License	Creative Commons Attribution-NonCommercial-ShareAlike
Author(s)	Code.org
Material Type	Online course
Media/Technical Format	Text/HTML
Short description of the material	The course covers topics such as problem solving, programming, physical computing, user-centered design, and data, artificial intelligence, and machine learning, while inspiring students as they build their own websites, apps, games, and physical computing devices.
What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches	A lot of materials and courses are offered, some of which are specially designed for high school students. The methods and approaches should be further analyzed to see which ones could be useful (i.g. CSA curriculum incorporates culturally responsive and equitable teaching strategies designed to invite, engage, and empower a rich diversity of students, https://code.org/educate/csa)
Need for additional analysis?	YES
Lectures	Full course catalog (English only) for students 14-18+. **CS Discoveries and CS Principles**. Both are designed to broaden participation in computer science. **Video Library** - These videos can be used in any CS course to support learning. https://studio.code.org/courses
Exercises	Various, from creating games to coding AI. **Hour of Code** - Hour of Code tutorials (Minecraft, Frozen, Flappy bird, Star Wars etc.) If you don't have time for a full length course, try a one-hour tutorial designed for all ages and 45 languages. 1/10 - 18/12 each year. Join. **Teachers guides** for various lectures, e.g. Lesson 1: Coding a Simulation
Tools	Tools for middle and high school (English only): **App Lab** is a programming environment where you can make simple apps. **Game Lab** is a programming environment where you can make simple animations and games with objects and characters that interact with each other. **CS Journeys** - Bring CS to life and help students make real world connections to what they're learning. **Widgets -** Students can explore concepts from our CS Principles course hands-on using these tools.
Evaluation	Dynamic evaluations to test knowledge while test-playing.
Other important aspects	Game-based learning, great motivation. But not related to OOP.
Final decision (what to use for OOP4FUN)	Great examples of using interesting innovative teaching methods for video game development.

Climate Solutions Challenge (OER Commons)

Title of learning/teaching material with the URL on the repository	Climate Solutions Challenge
Direct URL to the learning material	https://www.oercommons.org/courseware/lesson/75955/overview

Date added	2020.
License	Creative Commons Attribution
Author(s)	Brian Henning
Material Type	Activity/Lab, Lesson Plan
Media/Technical Format	Downloadable docs
Short description of the material	Project activity which focuses on students' familiarity with action items to reduce their carbon footprint.
What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches	Idea for project activity.
Need for additional analysis?	YES
Lectures	There are no lectures defined. It contains a more workshop-oriented approach.
Exercises	Exercises are defined in details with steps and approximate duration. Materials look like a detailed written preparation for a teaching session.
Tools	There is a special tool related to climate changes which is incorporated in the teaching/learning process.
Evaluation	Not defined.
Other important aspects	Maybe we can use the idea of a climate-challenged topic.
Final decision (what to use for OOP4FUN)	YES, as we will also deliver something similar to a learning/teaching scenario, maybe not in so much detail.

Five (5) materials were identified as **potentially useful** in terms of material structure or application idea.

Alice Programming Language (MERLOT)

Title of learning/teaching material with the URL on the repository	Alice Programming Language
Direct URL to the learning material	http://www.alice.org/ http://www.alice.org/resources/
Date: added	2005
Date: modified	2020
Author(s)	Stage 3 Research Group, Carnegie Mellon University
Material Type	Collection
Media/Technical Format	Web site, applications Alice2 and Alice3
Short description of the material	Alice is a 3d graphics programming environment intended to be a gentle first introduction to students ranging from 6th grade to college… Using programming concepts and structures, students build 3D virtual worlds that are often compelling. In the Alice project, they acknowledge that capturing someone's attention is a prerequisite to teaching them. Even a student's first programs can contain storytelling and game aspects, making the process of writing a program much more compelling, especially for female students.
What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches	Examples of resources: How to. Lessons, Exercises & Projects, Audio Library, Curriculum, Textbooks. Very high quality materials. There is probably also about o-o programming, so it should be reviewed in more detail. When searched with the keyword "object", it gives out several materials.
Lectures	All lectures related to programming in Alice applications (2 and 3). Content: Lessons, Skills Included in this Lesson, Exercises and Projects. Materials include ready to use presentation slides. Lessons materials are traditional content delivery oriented. Materials are linked with How Tos, Exercises and Projects.
Exercises	Exercises and Projects - Tutorials, Scene Building, Animation... all related to Alice Materials contain detailed instructions for facilitating lectures and exercises step by step.
Tools	The tool is available for free and all materials are related to the use of Alice tool.
Evaluation	Mentioned, but not defined in details.
Other important aspects	There is a link to a google sheet containing a detailed curriculum on teaching programming in Alice. Could be useful for OOP4Fun. http://www.alice.org/resources/curriculum/building-an-alice-curriculum/ **Google spreadsheet with the curriculum structure:** https://docs.google.com/spreadsheets/d/1Os-28thAt29jfa2LttK9w5H4nWQ1p46P3ln5aW7Uvo8/edit#gid=0.

Final decision (what to use for OOP4FUN)	Curriculum builder could be useful. Structure of lessons as well.

Teach Yourself C++ in 21 Days (MERLOT)

Title of learning/teaching material with the URL on the repository	Teach Yourself C++ in 21 Days
Direct URL to the learning material	https://www.angelfire.com/art2/ebooks/teachyourselfcplusplusin21days.pdf
Date: added	2004
Date: modified	2018
Author(s)	Greg Wiegand Sams Publishing
Material Type	Textual online course
Media/Technical Format	PDF
Short description of the material	Introductory C++ tutorial designed to give the user understanding of how this language works.
What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches	The material is really large (772 pages), but mostly it deals with detailed clarification of the characteristics of the C++ language. What could be analyzed in more detail is the chapter on OOP (analysis and design) on page 551. It explains the choices of development methodologies, ways of analyzing problems and designing solutions in an OOP perspective.
Lectures	Good source for teachers related to content and basics of OOP.
Exercises	Yes, examples of exercises (programming code).
Tools	No.
Evaluation	Q&A, Quiz (textual questions)
Other important aspects	-
Final decision (what to use for OOP4FUN)	Valuable resource only if C++ will be used in OOP4Fun. Without innovative teaching methods.

Data Discovery (MERLOT)

Title of learning/teaching material with the URL on the repository	**Data Discovery**
Direct URL to the learning material	http://www.searchingspot.com/datadiscovery/
Date: added	2012
Date: modified	2019
Author(s)	Mark Wenning
Material Type	Lekcije
Media/Technical Format	Website

Short description of the material	The lessons are designed to engage students with real-world data relevant to content taught in middle school and high school science courses. The Python lessons guide students in computational thinking to create simple programs to manipulate data. The lessons also provide students (and teachers) with instructions and guidance in the use of these technologies. Worksheets and supporting files are linked to from links at the top of each lesson webpage and from the downloads page.
What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches	The lessons are well designed, but they won't be very helpful for creating materials related to OOP and computer games. Inside the repository there is a link to additional materials on the page: https://learn.iste.org/d2l/lor/search/search_results.d2l?ou=6606&lrepos=1006 . This is the Exploring Computational Thinking repository, where there are ready-made Python tasks grouped according to the area or age for which they are intended.
Lectures	Lectures have an interdisciplinary approach combining informatics and other school subjects such as physics, chemists and other. **Cons**: Lectures focus on school knowledge while neglecting real life situations and motivation aspects.
Exercises	Exercises are based on worksheets for students (often to work in pairs) with clear structure, they include complex assignments which are divided into subtasks. **Cons**: Negative is that worksheets do not include relevant context for the assignments nor do they support complex learning approaches such as problem based learning, inquiry based learning and similar.
Tools	Assignments often use spreadsheets and python. Instructions on required tools are given.
Evaluation	Although concrete examples for grading are not presented, spreadsheets can be easily adopted for evaluation purposes.
Other important aspects	For Croatian teachers these concepts should be well known. However, the context should be added to the materials. On the other hand these assignments should be abstracted so the student reasons the tasks to be performed and solutions on his own.
Final decision (what to use for OOP4FUN)	The materials if improved as explained in 'other important aspects' could be of interest for us.

Lightbot Hour of Code (MERLOT)

Title of learning/teaching material with the URL on the repository	Lightbot Hour of Code
Direct URL to the learning material	https://lightbot.com/hour-of-code.html
Date: added	2016
Date: modified	2018
Author(s)	Danny Yaroslavski
Material Type	e-course
Media/Technical Format	Game
Short description of the material	It is a "game" or "puzzle" in which the user programs a robot using a visual, token-based programming language (icons).

What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches	It is not a learning material, but a puzzle game based on programming, the purpose of which is to convey the logic of programming to students while playing. Students learn the principles of programming (sequence, overloading, loops...). It is an innovative approach to teaching, so it would be useful to review the materials in more detail.
Lectures	Just info about principles of the game at the page How does Lightbot teach programming?
Exercises	None.
Tools	Lightbot for Win/Mac/Android/Ios. Paid application (free for MacOS).
Evaluation	None.
Other important aspects	The tool has been around for more than decade (written originally in flesh) and it has always been an interesting (almost brilliant) idea to teach basic concepts of programming. It would be great if we could create a game that would teach programming as well.
Final decision (what to use for OOP4FUN)	Great idea to build an application similar to this one using Greenfoot.

Simulating the Bug (OER Commons)

Title of learning/teaching material with the URL on the repository	Simulating the Bug
Direct URL to the learning material	https://www.oercommons.org/courses/simulating-the-bug/view
Date: added	18.9.2014
License	Educational Use Permitted
Author(s)	Douglas Bertelsen
Material Type	Activity/Lab
Media/Technical Format	Text/HTML
Short description of the material	Students modify a provided App Inventor code to design their own diseases. This serves as the evolution step in the software/systems design process.
What could be used from the material that contributes to PR2 goal: create learning designs with new teaching approaches	It has a contemporary idea of disease transmission and additional resources, including application with code.
Lectures	There are no lectures defined. It contains practical assignments and apk files.
Exercises	Materials provide basic input for the use of the tool (App Inventor). They focus on practical and interesting concepts of disease transmission and make an app for tracking.
Tools	https://appinventor.mit.edu/
Evaluation	Not available. It's meant to be used as a live exercise with a teacher.

Other important aspects	If App Inventor (MIT) would be used, then we could do the same thing this material did: in the beginning of the materials, place down all the blocks that are to be used.
Final decision (what to use for OOP4FUN)	Look at the **App inventor** tool.

3.4. Conclusions on the teaching/learning materials

Preliminary analysis of the MERLOT database showed that not many teaching/learning materials are available in the database MERLOT related to programming and specifically to object-oriented programming. The next step of the team was to investigate 44 teaching/learning materials found in the OER Commons search results and in the Croatian national repository of open educational resources Edutorij. The criteria for searching OER Commons (and MERLOT) were the following: keywords "programming" or "game-based programming", filtered by primary user (teacher), education level (high school), subject area (computer science) and material type (activity/lab, homework/assignment, interactive, lesson plan, module, teaching/learning strategy).

After identifying 19 educational materials from aforementioned repositories relevant for PR2 goal (create learning designs with new teaching approaches), additional deeper analysis of materials has been performed by two team members: one with the main background in programming and the other with the main background in didactics and/or e-learning methods. The focus of the deeper analysis was on the **a)** topic of object-oriented programming, and **b**) innovative pedagogical practices, methods and tools such as problem based learning, inquiry-based learning, game-based learning, flipped classroom, project based learning, peer learning, etc… Materials were explored from the perspective of the lectures, exercises, tools and evaluation.

Lessons learned from the second round analysis of relevant educational resources, that are worth implementing in the OOP4Fun context, are the following:

- There are not many open educational resources related to object-oriented programming (OOP). Those that are published refer to specific programming platforms, e.g. Alice, VB.Net, Greenfoot, and introduce some OOP concepts. Examples from these platforms could be used as an idea for learning designs in OOP4Fun PR2.
- Some educational materials introduce a specific OOP application, e.g. MIT App Inventor tool, that could be used for various topics by demonstrating OOP concepts.
- Several materials provide examples of **teaching scenarios** with inclusion of contemporary pedagogical methods. Those materials **could be used as guidelines** in creating learning designs for OOP4Fun.
- There are a few materials with presentations of a game or game-based learning that could be used in a similar way in OOP4Fun: e.g. creating a game and introducing the OOP concepts in the process of game development.
- Additional review of the **teaching resources available in Greenfoot.org** is needed (accessible after registration to the teacher's community) since the platform is proposed in the OOP4Fun project application.
- Most of the **analyzed materials do not provide a holistic approach** but usually place more focus on programming (how to code) or pedagogical aspects (novel ideas, contemporary pedagogical approaches, context relevant to real life situations, etc.). Our recommendation is to combine positive aspects from the analyzed materials into integrated solutions which incorporate both aspects.

- Materials mostly do not cover **evaluation aspects** or focus on traditional approaches (quizzes and similar). This is the field which **will require a novel approach**.
- Most materials do not make strict separation between exercises and lectures (instructions), which is a positive aspect and should be followed in the OOP4Fun. Most places focus on students' activity, while some do maintain a rather traditional approach such as exercises. There are several examples which focus on project activities or similar aspects which provide a more novel approach to learning.
- Many analyzed sources provide relevant materials for teachers - how to code, but do not provide contextual value to the presented exercises (like real life situations, motivation, application in different scenarios, etc.)
- Most of the analyzed materials are based on clear tasks for students in which the problem is already identified. This is not in line with real life situations and the nature of programming where one usually needs to identify what is the problem. Our suggestion is to **place emphasis on problem-based learning and project based learning** to support students' deeper learning.

We would like to give the credits and thanks to our partners from UNIZA and UNIBG who joined the efforts of the FOI team and reviewed all the materials about programming extracted from the educational repositories.

Also the university of Zilina, as participants in the National program IT Academy in Slovakia, have created the materials for OOP focused on students by means of utilizing the possibilities of the Greenfoot tool. Given the importance of these materials and the connected workshops that were executed, the analysis including the feedback from students and teachers who have already used the materials in their practice is given in the separate chapter in this document. The final conclusion supported with the feedback from teachers justifies that the Greenfoot as a tool should be used in OOP4fun project application.

4. Literature Review

acizmesi@foi.hr, dperas@foi.hr, zstapic@foi.hr

4.1. Introduction

In the 21st century, programming has emerged as a crucial skill, akin to fundamental abilities like reading and writing. Despite its paramount importance in contemporary workplaces, the methods employed to teach programming are often outdated, not aligning with the needs of digital-native learners. As a consequence, the demand for programming skills has surged, making it an integral aspect of formal education. However, mastering programming, especially for beginners, remains a formidable challenge, resulting in high dropout rates at universities and diminished student motivation. Recognized as one of the most difficult subjects, programming poses challenges for both learners and educators.

Research conducted over the past decades sheds light on the multifaceted difficulties faced by students and teachers in programming education. Notably, the diverse backgrounds of both educators and learners significantly influence their grasp of programming concepts. Beyond syntax and semantics, programming demands a comprehensive skill set. Various factors contribute to student struggles, including inflexible learning approaches, language barriers, inadequate math skills, lack of learning strategies, and educators' insufficient subject knowledge.

Early encounters with abstract programming concepts often prove daunting for many students, particularly in the context of traditional teaching methods that may disengage today's Generation Z. Addressing these challenges necessitates an exploration of the backgrounds and competencies of teachers and students as learners. This section delves into this exploration, presenting findings from a tertiary study conducted to enhance the effectiveness of programming courses. The subsequent sections outline the research problem, questions, protocol for the Systematic Literature Review (SLR), its implementation, and the results, followed by a discussion and concluding with key insights[1].

4.2. Conducting the analysis

Literature review was conducted at the end of August 2022 and included two relevant scientific bases for Information Science education research: Scopus and Web of Science (WoS).

Search Query in **Scopus** was following: **(TITLE-ABS-KEY (programming) AND TITLE-ABS-KEY (teach*) AND (TITLE-ABS-KEY (method) OR TITLE-ABS-KEY (approach)))** and initial results revealed 11 020 relevant documents. After that, the filter option was applied to include only review articles Scopus was used and resulted in 179 papers. Also, we have limited the search to relevant areas and included only documents from Social Science, Computer Science and Engineering and output of this was 121 documents in total. Next step was to read the title and the abstract of each paper and determine whether it is suitable for the purposes of project task analysis. By applying inclusion and exclusion criteria (Table 1), 21 papers were relevant, but 17 of them were available to download. After reading the papers, two independent researchers excluded an additional 10 papers which resulted in a total of **7 papers from Scopus suitable for literature review.**

[1] The research presented was published as a: Čižmešija, A., Peras, D., Stapić, Z., “Key competences and skills for teaching and learning programming: a tertiary study”, ICERI2023, Seville, Spain, 2023

Second online scientific database that was searched was **Web of Science (WoS)**. Query in WoS was following: ***Programming (Title) and teach* (Title)***. Other key words were excluded to optimize the search to the possibilities of researchers involved in the process. Initial search resulted with 9,469 results from Web of Science Core Collection. After that filter option/limitation to only review arciaels was applied resulting in 106 documents. Another step was taken – we included only relevant areas related to Education and Programming which gave us 51 papers that were carefully examined by reading their title and abstract. In the end, **5 papers from Scopus were suitable** for literature review and both independent researchers decided to keep them.

To get more possible relevant papers, **additional search** of Google Scholar was performed and resulted with 3 papers that were included in literature review. **Finally, 15 research papers in total were the starting point for detailed literature review.**

Table 1. Criteria used to include papers in literature review

Inclusion criteria	Exclusion criteria
Paper potentially deals with skills and competences of teachers or previous background knowledge of students related to programming	Document in the form of magazine article
Paper describes novel and innovative teaching methods for programming (+tools) and gives guidelines for future investigation of programming in education	Paper is not focused on programming in high school or higher education context
Paper is literature review or systematic literature review	Paper is thematizing programming in work-context

4.3. Literature review results

In the following section, selected papers are analyzed and relevant information is presented in line with the approaches used in other analysis for the purposes of PR2 so the final conclusions and outputs could be made in a uniform manner.

As it is shown in figure 5, 4 of the 15 papers included in the literature review were published 2017,. following by 2021. In which 3 papers were presented. By the distribution of relevant papers from year to year, we can see that topics related to exploring programming from various standpoints are novel and relevant, and therefore a very useful research and scientific area.

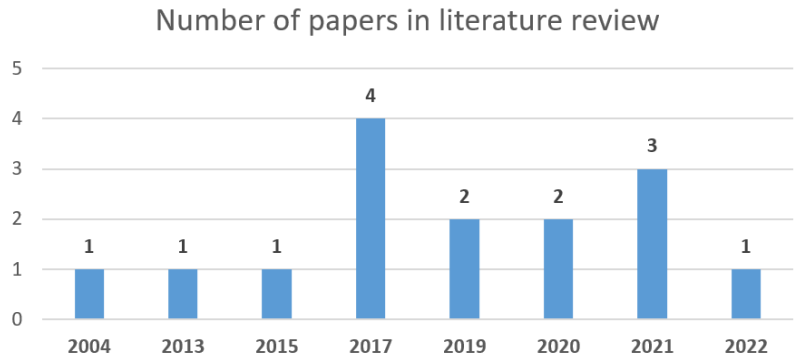


Figure 5. Number of papers in literature review by years

Table 2. shows an overview of the relevant papers covered by the literature review, which includes the names of authors, year, title, method used and the number of included works or included respondents. It is evident from the Table and paper titles that in our review the dominantly included literature reviews are related to programming in education in general, and only one paper deals with more detailed concepts relevant to object-oriented programming(OOP). Due to the lack of systematic works that will focus on OOP, the literature review included papers that will cover a broader research topic in order to answer these sub-research questions of our review:

- Which skills and competences teachers need to successfully teach programming?
- Which skills or previous background knowledge should students have to learn programming efficiently?
- Which innovative methods teachers use to successfully teach programming?
- Which tools can be used to successfully teach students programming?
- What are the main guidelines for future investigation of programming in education?

Table 2. List of paper included in literature review for project purposes

Authors and year	Title	Methodology	Number of analyzed papers/subjects	Level of education
(Medeiros, Ramalho, & Falcao, 2019)	A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education	Systematic Literature review	89 papers	Higher education
(Jawad & Tout, 2021)	Gamifying Computer Science Education for Z Generation	Pre-test-post-test experimental methodology	65 high school students	High school education
(Perugini, 2019)	Emerging languages: An alternative approach to	Case study	6 high school students	High school education

	teaching programming languages			
(Jenki & Ademoye, 2011)	Can Individual code reviews improve solo programming on an introductory course?	Literature review Experiment (pilot and follow up studies)	28 papers, 34 students	Higher education
(Yulianto, Prabowo, & Meyliana, 2017)	Effective Digital Contents for Computer Programming Learning: A Systematic Literature Review	Systematic literature review (2006 - 2017)	25 papers	Not specified
(Qian & Lehman, 2017)	Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review	Literature review	100 papers	High school education
(Hundhausen, Agrawal, & Agarwal, 2013)	Talking about Code: Integrating Pedagogical Code Reviews into Early Computing Courses	Mixed-method empirical study	21 students	Higher education
(João Henrique Berssanette & de Francisco, 2021a)	Active Learning in the Context of the Teaching/Learning of Computer Programming: A Systematic Review	Systematic literature review (2014 - 2019)	38 studies	Higher education
(Joao Henrique Berssanette & de Francisco, 2021)	Cognitive Load Theory in the Context of Teaching and Learning Computer Programming: A Systematic Literature Review	Systematic literature review (2010 - 2020)	33 papers	Higher education
(J. P. da Silva & Silveira, 2020)	A Systematic Review on Open Educational Games for Programming Learning and Teaching	Systematic Literature survey	12 studies	Not specified
(Abbasi, Kazi, & Khowaja, 2018)	A systematic Review of Learning Object Oriented Programming through Serious Games and Programming Approaches	Systematic literature review	15 studies	Higher education

(De Assis Mota, Mota, & Morelato, 2004)	Teaching Power Engineering Basics Using Advanced Web Technologies and Problem-Based Learning Environment	Case study + survey (student's feedback)	n/a	Higher education
(Luxton-Reilly et al., 2018)	Introductory Programming: A Systematic Literature Review	Systematic literature review	1666 papers	Higher education
(Hendrik & Hamzah, 2020)	Flipped Classroom in Programming Course: A Systematic Literature Review	Systematic literature review	32 papers	Higher education
(L. Silva, Mendes, & Gomes, 2020)	Computer-supported Collaborative Learning in Programming Education: A Systematic Literature Review	Systematic literature review	27 papers	Not specified

The papers included in the literature review covered a wide range of topics (for example: employing flipped classrooms in programming courses (Hendrik & Hamzah, 2020), open educational games for programming learning and teaching (J. P. da Silva & Silveira, 2020)), which is useful for understanding the problem area related to programming from multiple perspectives. This ensured us a good starting position because we were actually doing a meta analysis of existing knowledge and how to apply it to solve our identified problems and design methods related to teaching object-oriented programming in a fun way.

Most of the authors (9/15, 60%) used **Systematic literature review** as a method which ensured us a good starting position because we were actually doing a meta analysis of existing knowledge and how to apply it to solve our own problems and design methods that will teach object-oriented programming in a fun way. 2 of the papers used literature review and the rest of the papers used experiments or case study approaches (or its' combination) in conducting research. The most detail systematic literature review was carried out by (Luxton-Reilly et al., 2018) in which 1666 papers classified in 4 groups: *(1) Student* - student learning, underrepresented groups, student attitudes, student behavior, student engagement, student ability, the student experience, code reading, tracing, writing, and debugging, *(2) Teaching* - teaching tools, pedagogical approaches, theories of learning, infrastructure, *(3) Curriculum* - competencies, programming languages, paradigms, *(4) Assessment*- assessment tools, approaches to assessment, feedback on assessment, academic integrity.

Regarding the level of education in papers (Table 2) we recognized two as important: higher education and high school education. Most of the studies are dominantly conducted **at university level (60% of them)**, but some of those papers also included findings from high school or secondary school education (for example:(Hendrik & Hamzah, 2020),(João Henrique Berssanette & de Francisco, 2021a)). 3 of the analyzed papers ((Qian & Lehman,

2017),(Jenki & Ademoye, 2011),(Medeiros, Ramalho, & Falcao, 2019)) are focused on high scholls level, and the rest 3 do not explicitly state the level of education of students. Although authors in their work do not specify the education level on which research was conducted, they in the conclusion **pointed out the lack of the scientific papers dealing with open educational games to learn programming in higher education** (J. P. da Silva & Silveira, 2020). Therefore, research and practical opportunities still exist to apply different teaching methods that will be focused more on using games in teaching programming, especially OOP.

4.3.1. Summary of teacher skills and competences

Since the teaching of programming is done in the line with the use of various communication and collaboration tools dominantly in computer-mediated learning context, one pedagogical methodology that is indicated as appropriate is constructivism. In line with the fundamental concepts of constructivism, the role of the teacher in the process of programming knowledge delivery should differ from the classical tutor role and direct teaching. Teachers should motivate and guide students to acquire knowledge the related knowledge mostly independently through oriented searching (De Assis Mota, Mota, & Morelato, 2004). Furthermore, use of appropriate tools, interactive applications that are embedded in the learning environment in class or in some percentage online are key supporting factors in the teaching process and later, successful students' learning of programming concepts.

Similar approach that requires a more effective role of the teacher and her/his engagement is **active teaching and learning**. Findings of recent literature synthesis indicates that main contributions of active approach derive from teachers' role in order to **motivate students and stimulate their interest for the programming course, increasing their satisfaction**, improving learning experience and consequently, leading to better performance and outcomes (Berssanette & de Francisco, 2021) and keeping them engaged in the class (Medeiros, Ramalho, & Falcao, 2019).

The base of every successful knowledge transfer from teacher to student(s) is teacher's **knowledge of the subject matter**, in this case, programming concepts (Qian & Lehman, 2017) and use creativity to present them on the level students will comprehend. Creativity is especially relevant for teaching generation Z by using gamification (Jawad & Tout, 2021) regarding their high expectations as well as teachers' good problem solving skills (Medeiros et al., 2019). Being able to **clearly convey the tasks** and expectations that students should fulfill in the class is often mentioned competence in our literature review. Teachers should provide **well specified instructions** for various individual or group activities like project and presentations (Perugini, 2019) using appropriate analogies, models, and metaphors (Qian & Lehman, 2017). Also, teachers should give **tasks of appropriate complexity** (Qian & Lehman, 2017). In programming, teachers should be able to train and guide students before, during and after the code inspection. Some suggested activities include encouraging regular discussions of best coding practices and solutions to issues and keeping students on track on specific tasks that are expected from them to accomplish (Hundhausen, Agrawal, & Agarwal, 2013).

Another important guideline that should be considered by investigated literature is **teacher's flexibility and individual approach** to student or group if needed (Medeiros et al., 2019).

This especially applies to **balancing the classroom timeline** (for example expanding if needed) and putting focus both on gifted and more skillful students, as well as providing additional help and guidance for weak students (Berssanette & de Francisco, 2021). Sp**ecifically in active learning it is important to enable the greater flexibility and customizability of the teaching/learning process**, let students learn at their own pace and focus on activities that will increase students' confidence (Berssanette & de Francisco, 2021).

Regarding skills and competences teachers need to teach programing successfully using appropriate methods, importance of **communication skills** is well-elaborated in analyze literature reviews (Medeiros et al., 2019) as well as **providing timely feedback** to students (Medeiros et al., 2019), (Perugini, 2019).

Programming skills are often referred as hard skills, but besides them, teachers should devote the same amount of effort for class activities focused on **developing student's soft skills** (for example: teamwork, motivation, problem solving, critical thinking) and providing environment for efficient teamwork and collaboration (Berssanette & de Francisco, 2021).

4.3.2. Student's previous skills and background knowledge

In literature synthesis, authors (Medeiros, Ramalho, & Falcao, 2019) group skills students need in two main categories: **(1) general skills and (2) Programming related skills.** In this paper, we also followed this approach. Summary of desirable previous background skills from our literature analysis is shown in Table 3.

Apart from general skills that closely refer to soft skills and programming related skills/previous knowledge, literature analysis of selected papers indicated the importance of student's motivation high, and real engagement to progress in learning how to code an well as interest in programming (João Henrique Berssanette & de Francisco, 2021),(Jawad & Tout, 2021), (da Silva & Silveira, 2020). As other desirable skills, authors mentioned student's ability of self-regulation, restraint ability, independence, and responsibility in learning (Hendrik & Hamzah, 2020).

Table 3. Categorization and summary of student's previous skills and background knowledge

General skills	• Critical thinking and discussion skills, creativity, time management, english skills (Medeiros et al., 2019) • Soft skills: communication, collaboration, team work, self-efficacy, peer learning, critical review (Hundhausen, Agrawal, & Agarwal, 2013)

Programming related skills	• Problem solving, abstraction ability, mathematical skills (Medeiros et al., 2019) • Algorithm skills, mathematics skills, problem solving, logical thinking (Yulianto, Prabowo, & Meyliana, 2017) • Syntactic knowledge, conceptual knowledge, strategic knowledge, natural language, math knowledge, patterns and strategies (Qian & Lehman, 2017) • Development of higher cognitive skills (João Henrique Berssanette & de Francisco, 2021), (Joao Henrique Berssanette & de Francisco, 2021) • High levels of abstraction involved in programming logic (da Silva & Silveira, 2020)

4.3.3. Summary of other important aspects and guidelines for future investigation

In general, it is crucial to connect and encourage information exchange between school and university teachers, with the involvement of policymakers who define curricula related to programming skills at all educational levels. This **unified approach would make high school students more ready and skillful for entry level at university programmes containing programming courses** (Medeiros, Ramalho, & Falcao, 2019). Some initiatives have been already made, but there is a lot of open space for example to practice the flipped classroom for programming in the K-12 education level (Hendrik & Hamzah, 2020).

Developing and en**hancing teachers' pedagogical content knowledge and ability to apply effective instructional methods and tools to help students to overcome problems with programming concepts in OOP** (for example their misconceptions about content) is crucial for the success of teaching introductory programming and better students performance (Qian & Lehman, 2017).

For learning and teaching of OOP concepts, **learning by creating the games** showed the significant effects for improving students problem solving skills and engaging them in a fun and entertaining environment (Abbasi, Kazi, & Khowaja, 2018). Students' intrinsic and extrinsic motivation for computer science subjects can be improved by implementing class activities and gaming code examples. If students are able to apply learned programming concepts in creative ways and their code is functional (runs without errors), teachers should try to employ learning tools that have social buttons (for example: like, share) in order to provide them a base for sharing their knowledge with peers. For generation Z this could be an effective approach because such tools have similar functionalities of commonly used social platforms like Instagram, TikTok or Facebook. The most effective elements in gamified learning are points, badges, and a leader board (Jawad & Tout, 2021).

Regarding suitable resources (digital contents) for teaching and learning programming, authors (Yulianto, Prabowo, & Meyliana, 2017) suggest to use **e-books and videos for improving knowledge and other forms like animation/simulation** and games are useful for boosting students' motivation for programming and course related activities .

Other guidelines suggest that the students have to be masters of their own learning time and do it at their own pace through asynchronous learning segments(De Assis Mota, Mota, & Morelato, 2004). Learning in an online environment which was well established practice during the COVID-19 pandemic could be effectively incorporated to the course contents, increasing the appeal for learning. This way of teaching can replace traditional in situ lectures and optimize students' time for learning programming. Another note goes in the direction of the teachers/universities that have to encourage students in making connections with the real life sector (for example provide an opportunity for professional internship in industry).

4.3.4. Innovative forms of instruction/knowledge transfer

Papers dealing with the forms of knowledge transfer were grouped into course orientation and teaching delivery. The course orientation category includes papers that describe the overall approach to the structure of the course. The teaching delivery category includes papers that describe techniques and methods that could enhance learning.

Several variations of course designs were suggested, such as **blended learning** (Henrique Berssanette & Carlos de Francisco, 2021; Yulianto et al., 2017), **learning-by-doing** (Medeiros et al., 2019), problem solving (Hendrik & Hamzah, 2021), collaborative problem solving (Luxton-Reilly et al., 2018), teamwork (Medeiros et al., 2019), **problem-based learning** (Hendrik & Hamzah, 2021; Henrique Berssanette & Carlos de Francisco, 2021; Mota et al., 2004), active learning (Henrique Berssanette & Carlos de Francisco, 2021; Medeiros et al., 2019; Perugini, 2019), **lab-based learning** (Medeiros et al., 2019), extreme apprenticeship (Medeiros et al., 2019), inverted and flipped classrooms (Henrique Berssanette & Carlos de Francisco, 2021; Luxton-Reilly et al., 2018; Yulianto et al., 2017).

Teachers have explored a variety of contexts that connect introductory programming with real-world applications. It has been argued that the relevance and real-world application of computational processes are demonstrated effectively by tangible computing devices such as a Sifteo Cube, LEGO MindStorms, iRobot Creates, CSbots, and the Raspberry Pi (Luxton-Reilly et al., 2018). Although such innovative contexts are seldom evaluated, there is some evidence that students enjoy using the devices, and this enjoyment may translate to greater participation and completion rates. Block-based programming environments for beginners were suggested to help develop computational thinking, since they are not tied to a professional programming language as taught in university courses (Medeiros et al., 2019). Type of games commonly developed were Mini Games and Simulations, Adventures, RPG, Exploration, Puzzles, Role playing games (RPG), and Robot Simulations (Luxton-Reilly et al., 2018).

Various methods were used to deliver the content (Henrique Berssanette & Carlos de Francisco, 2021; Jawad & Tout, 2021; Jenkins & Ademoye, 2012; Luxton-Reilly et al., 2018; Medeiros et al., 2019; Silva et al., 2020): live coding, games, game-themed assignments and gamification; peer instruction, pair programming, peer review, electronic voting, pair programming, and group activities were used to deliver the content.

4.3.5. Methods and approaches

Lectures

The traditional teaching based on lectures has been questioned a lot. Part of the concern with the efficiency of traditional teaching stems from the teaching format that can often place students in a passive role so that they receive the knowledge transferred by the teacher from isolated facts, out of context, in addition to this knowledge being abstract with the significant possibility of being forgotten later. Flipped classroom teaching approach was suggested as most suitable for active learning/teaching because of (Henrique Berssanette & Carlos de Francisco, 2021): (i) increased learning performance; (ii) positive attitudes; (iii) increased engagement; (iv) more discussions; (v) enforced cooperative learning; and (vi) better learning habits.

Some of the basic instructional approaches and strategies suggested for lectures were to teach programming strategies and patterns explicitly to students, to use good program examples, and to develop a concept inventory (CI) by using emerging languages (Qian & Lehman, 2017).

One of the approaches proposed for coping with difficulties of understanding and mapping the basic concepts of OOP was Game-Based Learning (GBL) approach (Abbasi et al., 2017) (GBL). GBL provides close to real life scenarios in the engaging way. The investigated studies applied different programming approaches for OOP learning. Game first programming approach has been observed in most of the studies, followed by object, GUI, code and concept first approach.

Seminars/Laboratory exercises

Seminars/exercises should generally be designed to enable students to participate in the exercises (Jawad & Tout, 2021). Their inclusion in the creation of language synopses, the development of programming exercises, the creation of outline-style webpages should be encouraged (Perugini, 2019). The in-class activities suggested for the laboratory exercises are (Hendrik & Hamzah, 2021): hands-on-experience, problem solving, short briefings, quizzes, assignments with teacher assistance, and student's presentations of the emerging languages. Student collaboration can also be encouraged in form of pair-programming, and programming in groups with more than two students (Silva et al., 2020) .

Practical assignments

Students were mostly assigned graded homework, involving both conceptual and programming exercises, to evaluate their understanding of the concepts. Their final projects involved a software system, a formal paper discussing it, and an in-class presentation (Perugini, 2019).

Assessment

Various metrics were used to estimate the difficulty of code-writing tasks and to assess specific learning outcomes.

A number of tools have focused on giving students feedback on their programming process (Luxton-Reilly et al., 2018). Bluefix can be integrated into the BlueJ IDE, providing programming students with crowd-sourced error diagnosis and repair feedback. NoobLab is an online learning environment that attempts to provide an analogue of the traditional in-lab tutor-student dynamic that is specific to programming. PRAISE facilitates anonymous review and delivers prompt feedback from multiple sources including peer-generated feedback.

For assessing the learning outcomes, analysis of written exam papers, questionnaires, single-concept questions, multiple-choice exam questions, individual feedback, and self-assessment tools were used (Luxton-Reilly et al., 2018). (Online) questionnaires were mostly used for assessing enjoyment, usefulness, interest, engagement and simplification of concepts (Jawad & Tout, 2021; Jenkins & Ademoye, 2012). Step-by-step evaluation was also proposed (Hundhausen et al., 2013), consisting of pre-survey of attitudes, pre-test of course knowledge, post-survey of attitudes, PCR Exit Survey, and Post-test of programming knowledge (as part of final exam). Effective and cognitive level outcomes (apply, evaluate, synthesis, active participation, and willingness to learn) were measured in one paper (Abbasi et al., 2017). Final examination grading and passing ratio of the students were commonly used as evaluation metrics.

4.3.6. Tools

A variety of tools have been developed to support teaching and learning programming. The majority of papers present new tools, including environments and editors. Environments such as the Penumbra Eclipse plug-in, BlueJ, DrJava Eclipse Plug-in, and COALA seek to provide suitable pedagogic environments while leveraging the power of industry-standard IDEs. Calico can support a variety of languages, pedagogical contexts, and physical devices, while Jeroo can provide a smoother transition to Java or C++. Here are the environments and editors that have been reported, categorized by the languages they support (Luxton-Reilly et al., 2018):

- C: LearnCS!
- C++: CLIP
- Habanero Java: DrHJ
- Haskell: Helium
- Java: ALE, Java-based platform for developing 2-D Android games, BlueJ, COALA, CodeMage, Decaf, DrJava Eclipse Plug-in, ELP, Gild, JGrasp, Jigsaw, Penumbra
- Jeroo: Jeroo
- Karel++: objectKarel
- Pascal: VIPER
- Python: CodeSkulptor, PyBlocks, Pythy
- Multiple Languages: AgentSheets, Calico, CodeLab, CloudCoder

While most authors propose using one programming language in one course, some authors encourage the use of different emerging languages to achieve different purposes (Perugini, 2019): JavaScript for first-class and high-order functions, Python for closures and lazy evaluation (e.g., list/generator comprehensions), Perl for scoping options, Ruby for first-class continuations, Haskell for pattern matching, type systems, and lazy evaluation; and Racket for fundamental functional programming.

Different types of tools have been designed to help students learn programming in general: assessment systems, compiler assistants, program submission systems (providing minimal feedback), drill-and-practice systems, and intelligent tutoring systems (Luxton-Reilly et al., 2018).

Debugging tools (Decaf, Whyline), design tools (UML diagrams, flowcharts, pseudocode), visualization tools (Online Python Tutor, Greenfoot, UUhistle), graphical programming, and automated assessment systems (Luxton-Reilly et al., 2018; Qian & Lehman, 2017) have also been successfully utilized to help students learn programming.

Some of the popular serious games used for teaching OOP were (Abbasi et al., 2017): MUPPETS, WormChase, Breakout, Othello, Alice, GAPS, Simoo, and Sifteo cubes. In majority of studies game genre was not specified.

Tools that were used for making serious games are (Abbasi et al., 2017): GameMaker, MinimUML, Amiga, Ztech design, and Unity.3D

4.4. Conclusions on the literature review

Blended learning and flipped classrooms were recognized as the most dominant designs used to improve students' programming skill and motivation. The importance of fostering learning of core principles of programming by focusing on student learning rather than instructor teaching was emphasized. Various authors emphasized the importance of establishing the c**ontext and framework that will enable students to deconstruct and reason about, as well as the importance of providing ample scope for individual critical thought, design, and creativity**. The integration of multiple languages was also encouraged. Furthermore, **flipped classroom teachin**g approach was suggested as most suitable for active learning/teaching.

Games were widely used to motivate students to learn programming. Students liked the opportunity to be creative when supported by an appropriate framework. Evidence in the literature suggests that students enjoy the social interaction resulting from collaborative activities, and that working collaboratively has several benefits such as improved engagement, confidence, retention, and performance. **Three different ways** of learning through games were proposed: learning by playing games, learning by creating games and learning by using game related tools.

Various metrics were used to estimate the difficulty of code-writing tasks and to assess specific learning outcomes. Final examination grading and passing ratio of the students were commonly used as evaluation metrics.

A variety of tools have been developed to support teaching and learning programming. The majority of papers present new tools, including environments and editors. Different types of tools have been designed to help students learn programming in general: assessment systems, compiler assistants, program submission systems, drill-and-practice systems, and intelligent tutoring system. Debugging tools, design tools, visualization tools, graphical programming, and automated assessment systems have also been successfully utilized to help students learn programming.

5. Experiences from Students

zstapic@foi.hr, mmijac@foi.hr, mmatijevi@foi.hr, lmasnec@foi.hr

5.1. Introduction

In order to obtain information on good and bad experiences on teaching practices, approaches, and tools in programming and object-oriented programming-related courses, a semi-structured interview with final-year students was conducted. A total of 10 (ten) students from the University of Zagreb, Faculty of Organization and Informatics were interviewed. The aim of the interview was to obtain qualitative and quantitative data related to the focus of the interview. The interview focused on the final-year students from different study programs who have passed different programming and object-oriented related courses and now at the end of their journey can reflect on the teaching and learning approaches from different points of view.

5.2. Interview design

Before conducting the interview, we requested permission from the Committee for Ethical Matters (hr: Etičko povjerenstvo) at the Faculty of Organization and informatics. The request to approve this interview is attached in Figure 6 and the Approval from the Ethical Committee is presented in Figure 7. The detailed instructions for the interviewers are prepared in advance and have been submitted to the Committee for Ethical Matters. These instructions are also attached in the text below.

The students were familiarized and they gave their consent that the interview will be sound-recorded so the researchers could replay the interview if there will be any doubts in the data analysis phase. The students are also familiarized with the fact that all recordings will be deleted upon completion of this project result.

Also, special care was taken to avoid discussing any particular teacher or teachers directly or indirectly during the interview. All questions were placed by focusing on all programming courses and all teachers instead of addressing any particular course or teacher.

Izv. prof. dr. sc. Zlatko Stapić
Fakultet organizacije i informatike
Pavlinska 2
42000 Varaždin

Etičko povjerenstvo
Fakultet organizacije i informatike
Pavlinska 2
42000 Varaždin

Varaždin, 15.7.2022.

Predmet: Zamolba za mišljenje Etičkog povjerenstva u vezi provedbe istraživanja vezanog uz primjenu inovativnih metoda i pristupa u poučavanju objektno orijentiranog programiranja

Poštovani,

Fakultet organizacije i informatike sudjeluje u Erasmus+ projektu čiji je cilj identifikacija i svladavanje jaza između ishoda učenja srednjih škola te ulaznih vještina i znanja vezanih uz objektno orijentirano programiranje potrebnih na fakultetu. FOI-jev projektni tim zadužen je za izradu analize inovativnih metoda i pristupa poučavanju. Svrha ove analize je postavljanje temelja za izradu novog kolegija koji će učenike srednjih škola upoznati s osnovnim konceptima objektno orijentiranog programiranja kako bi ih bolje pripremio za fakultet i povećao njihovu motivaciju za upis STEM studijskih programa. U svrhu izrade analize provest ćemo intervju sa studentima viših godina Fakulteta organizacije i informatike koji su odslušali nekoliko predmeta vezanih uz (objektno orijentirano) programiranje. Cilj intervjua je prikupljanje iskustava studenata vezanih uz metode i pristupe korištene u poučavanju predmeta vezanih uz programiranje.
Prikupljene podatke ćemo anonimizirati na način da ne sadržavaju imena, prezimena, JMBAG-ove ni email adrese studenata, tj. bilo koje podatke kojima bi se moglo povezati studente s njihovim iskustvima. Podaci o nastavnicima također neće biti prikupljati, budući da iskustva studenata vezana uz nastavničke vještine i kompetencije nećemo prikupljati pojedinačno po predmetima niti po nastavnicima, već zbirno za sve predmete. Podaci koje ćemo tražiti tokom intervjua su:
spol, studijski program, godina upisa studija, godina studija, završena srednja škola, razina znanja programiranja stečena srednjoškolskim obrazovanjem, razina znanja objektno orijentiranog programiranja stečena srednjoškolskim obrazovanjem, motivacija i interes za programiranje, razina stečenih kompetencija programiranja, razina stečenih kompetencija objektno orijentiranog programiranja, naziv položenog predmeta (vezanog uz programiranje), doprinos znanju, ocjena iz položenog predmeta, nastavničke vještine, nastavničke kompetencije, korištene inovativne metode i alati, ostali bitni aspekti, mogućnost kontaktiranja.
Rezultati analize podataka dobivenih intervjuom koristit će se za potrebe Erasmus + projekta, te u znanstvenim radovima kojima ćemo u kasnijim fazama projekta predstaviti projektne rezultate. Svi korišteni podaci i rezultati bit će zbirni, dakle potpuno anonimni.

Molimo mišljenje Etičkog povjerenstva o prikupljanju gore navedenih podataka u spomenutom istraživanju te o mogućnosti korištenja tih rezultata za potrebe Erasmus+ projekta i u znanstvenim radovima. U slučaju planiranja uključivanja podataka u nekom drugom kontekstu, poslat će se nova zamolba.

Srdačan pozdrav,
Izv. prof. dr. sc. Zlatko Stapić

Prilozi:
1. Plan interview s uputama i pitanjima

Figure 6. Request to approve the interview

foi
SVEUČILIŠTE U ZAGREBU
FAKULTET
ORGANIZACIJE I
INFORMATIKE
VARAŽDIN

HR - 42 000 Varaždin
Pavlinska 2

tel: +385 42 390 800
fax: + 385 42 213 413

ured-dekana@foi.hr
www.foi.unizg.hr

ETIČKO POVJERENSTVO

KLASA: 602-11/22-25/1
URBROJ: 2186-62-14-22-26
Varaždin, 18. srpnja 2022.

Izv. prof. dr. sc. Zlatko Stapić

MIŠLJENJE ETIČKOG POVJERENSTVA

Etičko povjerenstvo Fakulteta organizacije i informatike Sveučilišta u Zagrebu razmotrilo je zamolbu izv. prof. dr. sc. Zlatka Stapića od dana 15. srpnja 2022. godine kojom se traži mišljenje Etičkog povjerenstva u vezi provedbe istraživanja vezanog uz primjenu inovativnih metoda i pristupa u proučavanju objektivno orijentiranog programiranja. Etičko povjerenstvo je zaključilo da je prikazano prikupljanje podataka u spomenutom istraživanju te korištenje tih rezultata za potrebe Erasmus + projekta i u znanstvenim radovima u skladu sa načelima etičnosti, da je etički prihvatljivo te da je u skladu sa smjernicama Etičkog kodeksa Sveučilišta u Zagrebu.

PREDSJEDNICA ETIČKOG POVJERENSTVA:

Prof. dr. sc. Ksenija Vuković

Dostavljeno:
1. Podnositelju zahtjeva
2. Arhiva

Figure 7. Approval from the Committee for Ethical Matters

The interviewers were two teachers from the University of Zagreb, both having several years of teaching programming-related courses. To avoid possible different interpretations of the results the interviewers had an initial meeting to discuss the questions, the flow of the interview, and all other aspects of the interview. The instructions for interviewers were prepared in advance and are copy-pasted here as follows:

Instructions to the interviewer

Note that the texts in blue are instructions for the interviewer (questions are written in black).

The answers must be written in the provided template (excel file).

All questions are optional.

The interview is to be voice recorded.

Introduction

a. Introduce yourself

b. Explain the purpose of the interview:
The University of Zagreb, Faculty of organization and informatics along with four other universities from Slovakia, Germany, the Check Republic, and Serbia, and five high schools from the same countries is running an Erasmus+ project which aims to identify and eliminate the gaps between high school learning outcomes and university required input skills and knowledge related to object-oriented programming. One of the outputs of the project will be an innovative high school course in games development which will introduce high school students to the basic concepts of object-oriented programming (in order to better prepare students for universities as well as to increase their motivation for enrolling in STEM study programs in general).
Thus, we are currently analyzing the innovative teaching methods and approaches that could be used in that new high school course.
As a university student, you have already completed several courses related to object-oriented programming as well as programming in general and with this interview, we would like to ask you about your experiences which are related to teaching methods.

c. Explain that the answers will be treated anonymously and that the interview will be **voice recorded**. Put the password of the candidate in excel and start recording by introducing the password.

Profile

1. Some personal information is needed to contextualize the answers regarding other responders. Gender, Study program, Year of enrollment, Year of study, High school profile… (The possible answers are in the template, please write the answers in the provided template – Excel file)
2. What was the level of knowledge in programming and object-oriented programming gained in your high school education? The possible answers (no knowledge, low knowledge, moderate knowledge, above average knowledge, high knowledge) are in the template, please write the answers in the provided template – Excel file)

3. How high was your motivation and interest in learning programming during your studies? The possible answers (no interest, low interest, moderate interest, above average interest, high competencies) are in the template, please write the answers in the provided template – Excel file)
4. Which level of competencies have you gained during your studies in terms of programming in general and object-oriented programming? The possible answers (no competencies, low competencies, intermediate competencies, above average competencies, high competencies) are in the template, please write the answers in the provided template – Excel file)
5. Which courses related to programming have you passed, how much has that course contributed to your knowledge, and what was your final grade? Please write the answers in the provided template)

Teaching skills and competences

Introduction: The skill is the ability to perform certain physical tasks or activities in the desired way, while the competencies are the broader term that includes the skills, knowledge and attributes that enable a person to perform effectively in a job or situation. Please ask questions before giving examples, and use examples only if necessary. Some examples of *skills* can be the use of ICT, development skills, communication skills, skills related to the use of development environment, etc, while the examples of *competencies* could be data modeling, methodological development, problem-solving approach, teamwork and collaboration, organization and planning, classroom management, facilitation and engagement, assessment and mentoring, flexibility and adaptability…

6. Related to programming courses (and maybe other courses) can you enumerate the skills that you can recall that teachers had and that, in your opinion, contributed to the overall course delivery (success)? (Answers are to be written in the excel file.)
7. Related to programming courses (and maybe other courses) can you enumerate the skills that you can recall that teachers were lacking and that, in your opinion, contributed to the overall course delivery (success)? (Answers are to be written in the excel file.)
8. Which skills according to your opinion teachers should have in order to successfully deliver a programming or object-oriented programming-related course? (Answers are to be written in the excel file.)
9. Related to programming courses (and maybe other courses), can you enumerate the competencies that you can recall that teachers had and that, in your opinion, contributed to the overall course delivery (success)? (Answers are to be written in the excel file.)
10. Related to programming courses (and maybe other courses), can you enumerate the competencies that you can recall that teachers were lacking and that, in your view, contributed to the overall course delivery (success)? (Answers are to be written in the excel file.)
11. Which competencies according to your opinion teachers should have in order to successfully deliver a programming or object-oriented programming-related course? (Answers are to be written in the excel file.)

Teaching practices, methods, and tools

Introduction: The course content could be delivered in different ways. There are many novel and modern teaching practices, methods, and tools that could be used to help students understand the knowledge, apply the knowledge, keep concentration, increase motivation etc. Please ask questions before giving examples, and use examples only if necessary. Examples of teaching practices and methods could be gamification, work-based learning, team-based learning, table-based learning, project-based learning, project-solving approach, flipped classroom, hybrid approach, blended learning, short quizzes, and home preparation. Examples of tools include both programming tools and didactic tools such as (Git/Github for versioning, Wiki or Confluence for documentation, Jira for teamwork and agile practices,

12. Related to programming courses (and maybe other courses), what was the organization of the lectures? Were there any innovative teaching methods applied? Which methods do you find useful and which are obsolete? (Answers are to be written in the excel file.)
13. Related to programming courses (and maybe other courses), what was the organization of the seminars or exercises? Were there any innovative teaching methods applied? Which method do you find useful and which are obsolete? (Answers are to be written in the excel file.)
14. Related to programming courses (and maybe other courses), what were the technologies and tools used? Were there any innovative tools used? Which tools do you find useful and which are obsolete? (Answers are to be written in the excel file.)
15. Related to programming courses (and maybe other courses), what were the techniques and practices used for evaluation? Were there any innovative techniques and practices used? Which evaluation methods do you find useful and which are obsolete? (Answers are to be written in the excel file.)

Closing the interview

16. Finally, are there any other elements of teaching or factors you think are important to be considered when teaching programming or object-oriented programming? Is there anything you wish to add?

Closing of interview: Thank you very much for your cooperation. It has been exciting to talk with you, and I am sure your help will be of great use in the study we are conducting. I wonder if I can contact you on a future occasion to request further clarification of information or any additional contribution to the project.

5.3. Conducting the interview

Interviews were strictly done in accordance with the GDPR and FOI's ethical commission. For every student, the following data were collected:

- Gender
- Current study program
- Year of university enrollment
- Current year of study
- Type of high school they attended
- High school programming knowledge
    - of programming
    - of OOP
- Motivation and interest in programming
- Gained competences
    - in programming
    - in OOP
- Passed programming courses, their knowledge contribution, and final grades
- Skills and competences which the teachers had, lacked, and should have had.
- Innovative practices, methods and tools used in lectures, exercises, and evaluations.
- Other important aspects of college education on FOI.

The data was collected during the time span of several days and we collected the data in anonymized form in google spreadsheet as presented in the following screenshot.

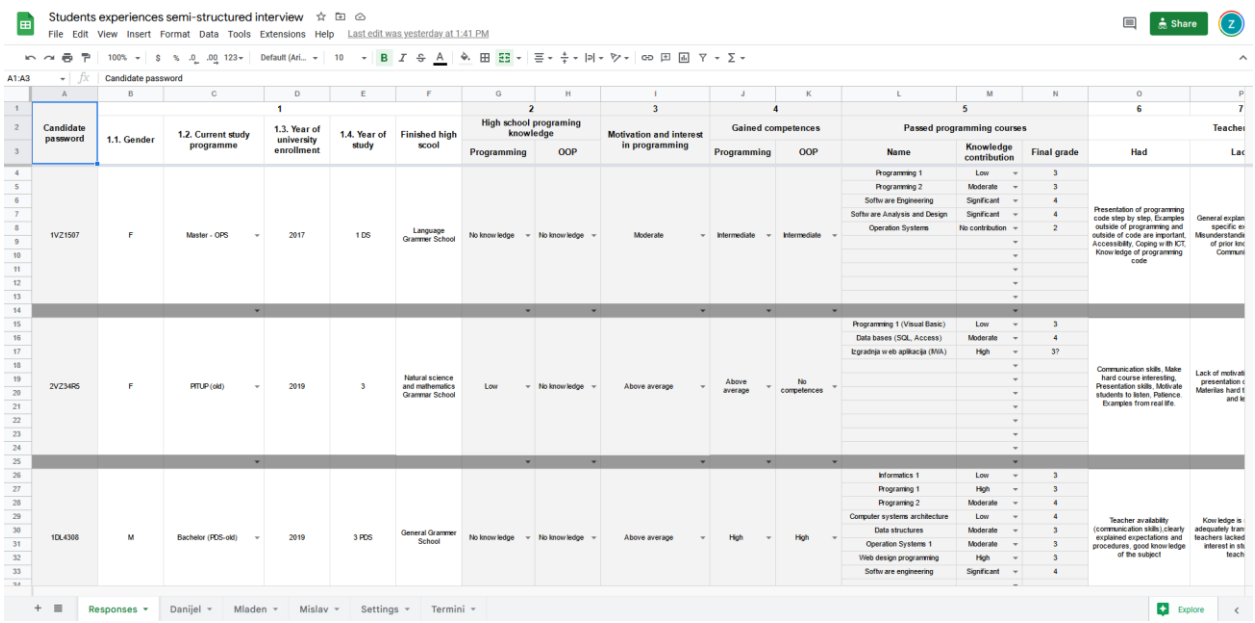
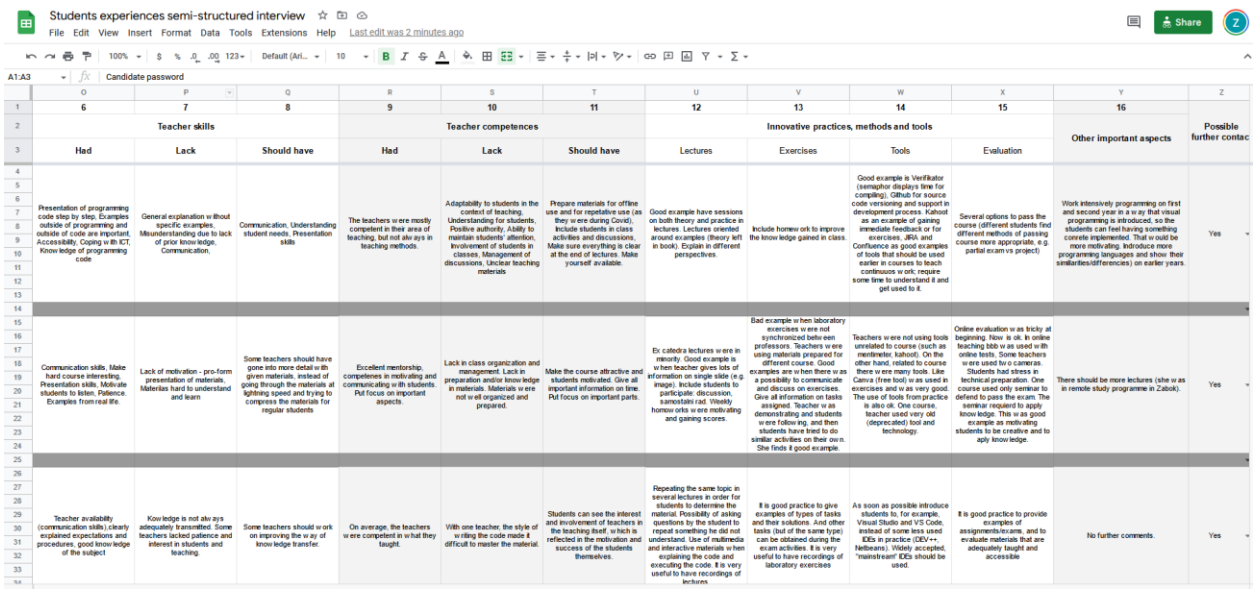


Figure 8. Spreadsheet with collected data (part 1)



Students experiences semi-structured interview

	6	7	8	9	10	11	12	13	14	15	16	
2	Teacher skills			Teacher competences			Innovative practices, methods and tools				Other important aspects	Possible further contac
3	Had	Lack	Should have	Had	Lack	Should have	Lectures	Exercises	Tools	Evaluation		
4												
5												
6												
7	Presentation of programming code step by step. Examples outside of programming and outside of code are important. Accessibility, Coping with ICT, Know ledge of programming code	General explanation w ithout specific examples, Misunderstanding due to lack of prior know ledge, Communication,	Communication, Understanding student needs, Presentation skills	The teachers w ere mostly competent in their area of teaching, but not alw ays in teaching methods.	Adaptability to students in the context of teaching, Understanding for students, Positive authority, Ability to maintain students' attention, Involvement of students in classes, Management of discussions, Unclear teaching materials	Prepare materials for offline use and for repetative use (as they w ere during Covid), Include students in class activities and discussions, Make sure everything is clear at the end of lectures. Make yourself available.	Good example have sessions on both theory and practice in lectures. Lectures oriented around examples (theory left in book). Explain in different perspectives.	Include homew ork to improve the know ledge gained in class.	Good example is Verifikator (semaphor displays time for compiling), Github for source code versioning and support in development process. Kahoot as an example of gaining immediate feedback or for exercises, JIRA and Confluence as good examples of tools that should be used earlier in courses to teach continuous w ork; require some time to understand it and get used to it.	Several options to pass the course (different students find different methods of passing course more appropriate, e.g. partial exam vs project)	Work intensively programming on first and second year in a w ay that visual programming is introduced, so the students can feel having something conrete implemented. That w ould be more motivating. Indroduce more programming languages and show their similarities/differencies) on earlier years.	Yes
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18	Communication skills, Make hard course interesting, Presentation skills, Motivate students to listen, Patience, Examples from real life.	Lack of motivation - pro-form presentation of materials, Materias hard to understand and learn	Some teachers should have gone into more detail w ith given materials, instead of going through the materials at lightning speed and trying to compress the materials for regular students	Excellent mentorship, competenes in motivating and communicating w ith students. Put focus on important aspects.	Lack in class organization and management. Lack in preparation and/or know ledge in materials. Materials w ere not w ell organized and prepared.	Make the course attractive and students motivated. Give all important information on time. Put focus on important parts.	Ex catedra lectures w ere in minority. Good example is w hen teacher gives lots of information on single slide (e.g image). Include students to participate: discussion, samostalni rad. Weekly homew orks w ere motivating and gaining scores.	Bad example w hen laboratory exercises w ere not synchronized betw een professors. Teachers w ere using materials prepared for different course. Good examples are w hen there w as a possibility to communicate and discuss on exercises. Give all information on tasks assigned. Teacher w as demonstrating and students w ere following, and then students have tried to do similar activities on their own. She finds it good example.	Teachers w ere not using tools unrelated to course (such as mentimeter, kahoot). On the other hand, related to course there w ere many tools. Like Canva (free tool) w as used in exercises and w as very good. The use of tools from practice is also ok. One course, teacher used very old (deprecated) tool and technology.	Online evaluation w as tricky at beginning. Now is ok. In online teaching bbb w as used w ith online tests. Some teachers w ere used tw o cameras. Students had stress in technical preparation. One course used only seminar to defend to pass the exam. The seminar required to apply know ledge. This w as good example as motivating students to be creative and to aply know ledge.	There should be more lectures (she w as in remote study programme in Zabok).	Yes
19												
20												
21												
22												
23												
24												
25												
26												
27												
28												
29												
30	Teacher availability (communication skills) clearly explained expectations and procedures, good know ledge of the subject	Kow ledge is not alw ays adequately transmitted. Some teachers lacked patience and interest in students and teaching.	Some teachers should w ork on improving the w ay of know ledge transfer.	On average, the teachers w ere competent in w hat they taught.	With one teacher, the style of w riting the code made it difficult to master the material.	Students can see the interest and involvement of teachers in the teaching itself, w hich is reflected in the motivation and success of the students themselves.	Repeating the same topic in several lectures in order for students to determine the material. Possibility of asking questions by the student to repeat something he did not understand. Use of multimedia and interactive materials w hen explaining the code and executing the code. It is very useful to have recordings of lectures	It is good practice to give examples of types of tasks and their solutions. And other tasks (but of the same type) can be obtained during the exam activities. It is very useful to have recordings of laboratory exercises	As soon as possible introduce students to, for example, Visual Studio and VS Code, instead of some less used IDEs in practice (DEV++, Netbeans). Widely accepted, "mainstream" IDEs should be used.	It is good practice to provide examples of assignments/exams, and to evaluate materials that are adequately taught and accessible	No further comments.	Yes

Figure 9. Spreadsheet with collected data (part 2)

5.4. Results of the analysis

A total of 4 female students and 7 male students were interviewed. 6 of these students are currently enrolled in postgraduate studies and the remaining 5 are currently in undergraduate studies. Two of the students on postgraduate studies are studying “Organization of business systems”, two are studying “Databases and bases of knowledge”, and two students are studying “Informational and programming engineering” study programs.

All students finished some sort of a Grammar school (Gymnasiums), except for two who have finished technical schools for computing. It seems as if high schools had little impact on differences of students’ programming knowledge prior to enrolling into college. For example, out of three students from natural science and mathematics grammar schools, one rated his pre-college programming knowledge as “low”, another one as “moderate”, and the third one as “high”. Three students from general grammar schools rated “no knowledge” in the same area, but a fourth one from the same type of school rated it “moderate”.

Students had completely different programming knowledge before attending college, but reported they’ve **gained between intermediate and high levels of knowledge of programming in college**. One student from the obsolete PITUP study reported he gained no competences in object-oriented programming (OOP). Other students reported they gained intermediate or high knowledge of OOP.

Students have also rated different course impacts on programming knowledge. Not all programming courses were mentioned, but only those which the students remembered to have had an impact or to have been important during their study. In order to perceive the impact of each programming course, course ratings were transformed into numbers and average was calculated. Transformation was done in the following order:

Table 4. Perceived course knowledge impact enumeration

Perceived course knowledge impact	Transformed number of rank
No contribution	0
Low	1
Moderate	2
Significant	3
High	4

Courses were then sorted primarily by their calculated average grade of knowledge contribution, and then by the number of mentions in the interviews. Average grades were rounded to one decimal. Top 10 courses from students' answers sorted this way are shown below:

Table 5. Coures everage grades

Course name	Course average grade	Course mentions
Advanced web technologies and services	4	2
Building of Web Applications	4	2
Web programming	4	1
Software Engineering	3.6	9
Web Design and Programming	3.6	5
Software Analysis and Design	3.3	4
Computer Graphics	3	1
Network 2	3	1
Physical modeling of databases	3	1

It's obvious that Software Engineering, Web Design and Programming, and Software Analysis and Design courses had a significant impact on most students (above the grade of 3 with most mentions). Some other courses well worth mentioning are Advanced web technologies, Building of Web Applications, and Web programming, all three with a grade of High.

It is worth noting that only some of these courses are object-oriented. Others were mentioned for their contribution to learning programming concepts.

Teacher skills were differently evaluated. Some commended how teachers presented programming or how interested they seemed, others praised communication skills and willingness to help. On the other hand, it was also noted that some teachers didn't show enthusiasm for their course or for the programming itself, or were unclear in their explanations. All in all, students' experiences differed so much that it was not possible to define which skills the teachers lacked, but the results clearly suggest which skills college teachers should have.

Students mostly noted that teachers should motivate students by explaining why a certain topic is being lectured, by giving interesting real-world stories related to the materials, by being available for additional explanations, etc. Teachers should organize and structure learning materials well and not get lost in them.

Innovative practices used in lectures that were mentioned are the demonstrations by examples, teamwork, discussions, short quizzes, usage of multimedia and interactive materials when explaining and executing the code, invited lectures or other display of real life examples.

There weren't many mentions of innovative practices used in exercises, but those mentioned were teamwork, recorded materials for learning at home, and demonstrations by teachers followed by students' further research.

Tools that have been proved useful for students while mastering the materials were mostly modern IDEs and platforms such as Visual Studio, Visual Studio Code, Github, Jira, Confluence etc. Unique tool "Verifikator" was mentioned. It's created by FOI teachers and used in the Programming 2 course. However, one student used it as a good example, and another held a grudge against it for being too opinionated.

Students believe they should be evaluated multiple times during the semester, not just by one final exam. Online evaluation was mentioned, but with its shortcomings. Term papers were mentioned as a good method for evaluating student's practical knowledge. Gamification was also mentioned as a good tool for motivating students to engage more with the course.

5.5. Conclusions on experiences from students

It seems that high school has little impact on programming knowledge. It's probably owing to the quality of teachers in schools and intrinsic motivation.

Furthermore, it seems study programmes don't have a very strong impact on programming knowledge. All students from university courses gained intermediate and high knowledge in OOP, except for one, from the obsolete PITUP study. However, PITUP study was not a university study, but a professional study not meant to delve too deep into programming concepts.

Software Engineering is rated by most students as a major contributor to programming knowledge on FOI. Software engineering as a course also happens to be heavily object-oriented. If OOP4Fun could move object-oriented programming into high schools, this course could explore more complex topics and contribute even more.

Students didn't universally praise or criticize the same teacher skills and competences. Some found that teachers had great communication skills and enthusiasm, others claimed the opposite. The reason behind that is probably the different set of teachers on courses and study programmes themselves. It's good that all qualities were enumerated as existent in all interviews, which means there are a number of quality teachers on FOI.

A thematic analysis of teacher skills and competences was conducted. Interview was designed to separately question skills and competences, but due to failure of students to separate the two while answering, they were combined in analysis with heavier emphasis on skills.

Qualitative answers were categorized according to similarity of terms used. A total of 10 categories were collected, each describing its own teacher quality:

- **Teacher availability** - how much teachers are open to helping students outside classroom
- **Knowledge of subject** - how well do teachers manage to explain the materials (e.g. with their own examples)
- **Communication skills** - how do teachers communicate with students (e.g. include them in discussions)
- **Presentation skills** - how well do teachers explain the materials in their lectures
- **Real life example** - how well do teachers connect real life examples with the lectures they're holding
- **Motivation** - how well do teachers motivate students in order to create interest for the subject
- **Self-motivation** - how much are teachers motivated to hold lectures
- **Flexibility** - how much are teachers flexible when confronted with students' demands
- **Up-to-date** - how modern are the teaching materials
- **Creativity** - using non-usual methods of teaching

Number of recurrences of categories was counted both in "Had" and "Should have" columns separately. The result of thematic analysis is shown in Figure 10.

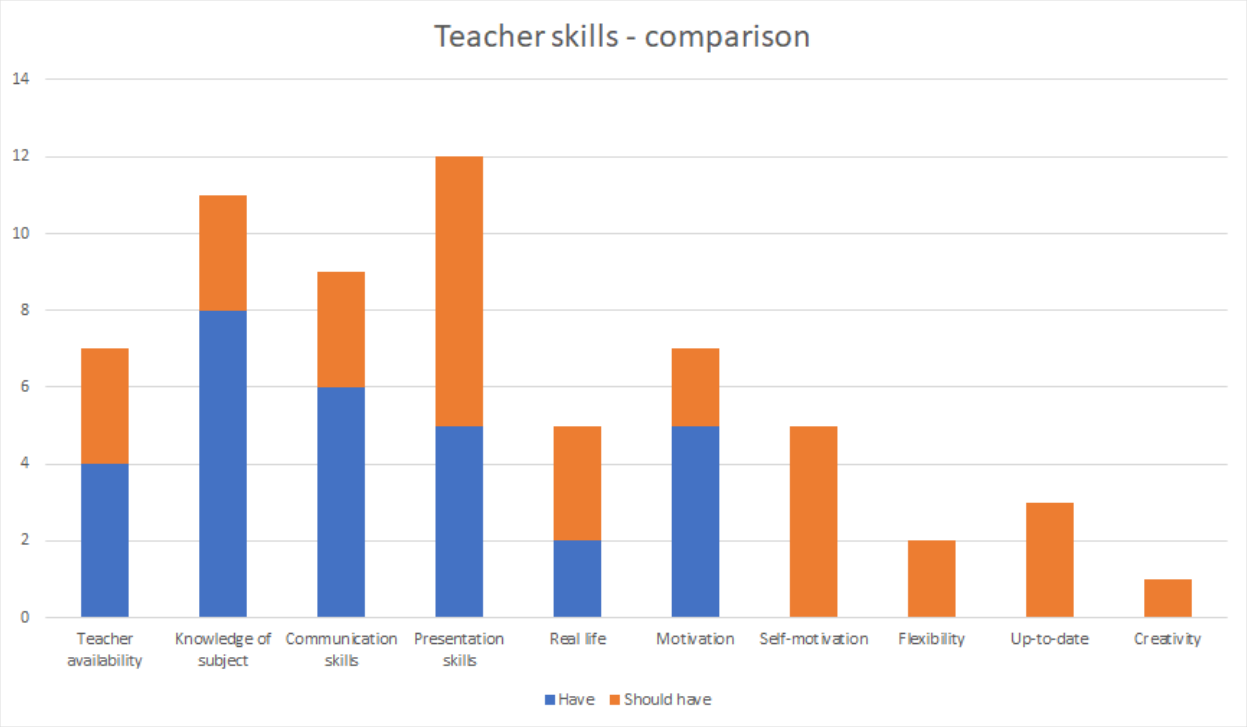


Figure 10. Teacher skills - comparison

It's apparent that students mostly commended FOI teachers for being knowledgeable about the subject they were lecturing and for having good communication skills. However, no one mentioned that teachers were self-motivated or that they had up-to-date materials. Unfortunately, those virtues were only noted as something teachers should have. It would also be good for college teachers in general to make sure they are flexible and creative enough to make learning their courses as enjoyable as possible.

Finally, students remarked that there should be more different tools and programming languages in earlier years of study. They believe programming concepts could be simplified using a visualization tool. Also, there was a complaint that OOP was not properly introduced because focus was partly on covering more basic concepts or narrower topics which would be more appropriate in separate courses.

6. Empirical results from UNIZA

UNIZA, mmijac@foi.hr

6.1. Introduction

The Faculty of Management Science and Informatics from UNIZA organized several workshops with the goal of making OOP more attractive among high school students. Two of the workshops were Active motivational workshops organized in 2020 as a part of the LOOP project (which received a grant from E-schools for future by Orange Foundation). The third workshop was a standalone workshop organized in 2022 as a Winter school of programming.

During the first workshop, 2nd year students had two days to develop two games (inspired by popular Icy tower and Tank battle games). Python programming language and PyGameZero module were used as tools. The second workshop was organized for 4th year students, who were given two days to develop a game which simulates the Enigma machine (for ciphering and deciphering messages from the 2nd World War). In this workshop students used Java programming language and one of the mainstream popular IDEs. During the third workshop students of the 3rd and 4th year of high school were given 3 days to develop a game using Java programming language and Greenfoot IDE.

The feedback obtained from participants after the workshops were done indicates that interesting topics, game-based learning and OOP are great combinations to motivate students to enter the world of programming and STEM field. In the following section more details about each individual workshop and their results is provided.

6.2. Active motivational workshops (Project LOOP)

Project LOOP was realized in 2020. It was supported with grant e-schools for future by Orange Foundation. Project was realized in cooperation with Gymnázium Viliama Paulinyho-Tótha in Martin, Slovakia (grammar school) [1]. Among others, one goal of the project was to make informatics more attractive among students as well as to enhance the knowledge of students of last year of study in the field of object oriented programming (OOP). Lectors prepared „active motivational workshops “, that were held online due to pandemic of COVID 19. The workshops were offered for students of grammar school and technical school:

- For the students of **second** year of study, PyGameZero module of Python programming language has been used. Students created within the two days two games (inspired by popular Icy tower and Tank battle games).
- Students of the **fourth** year of study have used programming language Java and proper IDE to study basic principles of OOP. Within two days they created application Enigma, capable to cipher and decipher messages from World War 2.
- As a side note, we add that there was also a workshop oriented for web technologies prepared for students in their third year of study. Since this workshop is out of scope of project OOP4fun, we have not included it into here presented analysis. However, the number of participants indicated, that interesting topic is very motivational factor for students.

Two of the aforementioned workshops were led by the team of researchers and experts from the Faculty of Managements Science and Informatics. Object first principle as well as game-

based development have been used. After the workshops were completed a feedback from students was collected, including the data on the type of school participants were attending, as well as participants opinion related to held workshops and lecturers.

The feedback was generally positive, i.e. students agreed on a professional but friendly approach of lectors, as well as on very interesting topics of workshops. Also, from the collected data, we can see that students of technical school had bigger motivation to attend the PyGameZero workshop (2nd year), while students of grammar school were more focused on the Enigma workshop (4th year). This might indicate that students of technical schools are very early accustomed to STEM oriented programs, while students of grammar schools still need a bit of push in the earlier stages of their higher education.

The overall results indicate that the correct approach has been chosen. Therefore, we consider educational materials for teachers on the base of light OOP and game-based development to be validated as a correct approach with potential to succeed in pedagogical praxis after pilot deployment. Using game-based development can be used with benefits to motivate students into studying STEM oriented fields. Moreover, we see that light OOP approach will be able to transfer to different programming languages or IDE (using this feedback, we conclude PyGameZero to be successfully tested). This result of analysis will be taken into consideration when creating outputs for respective project goals.

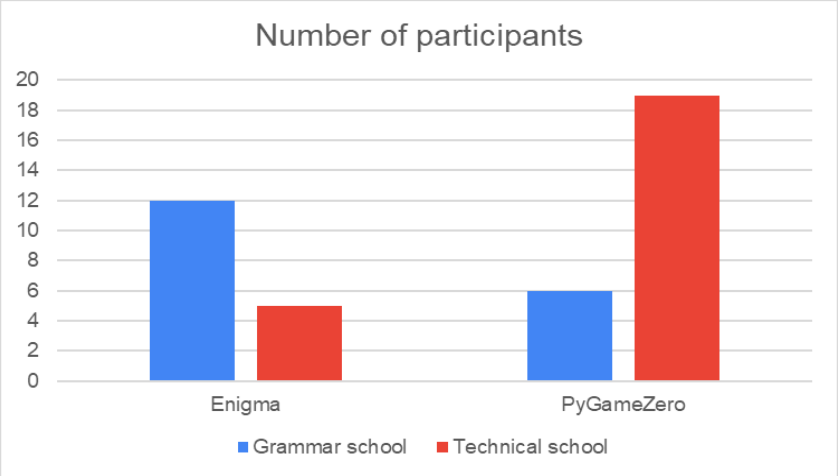


Figure 11. Number of participants on workshops

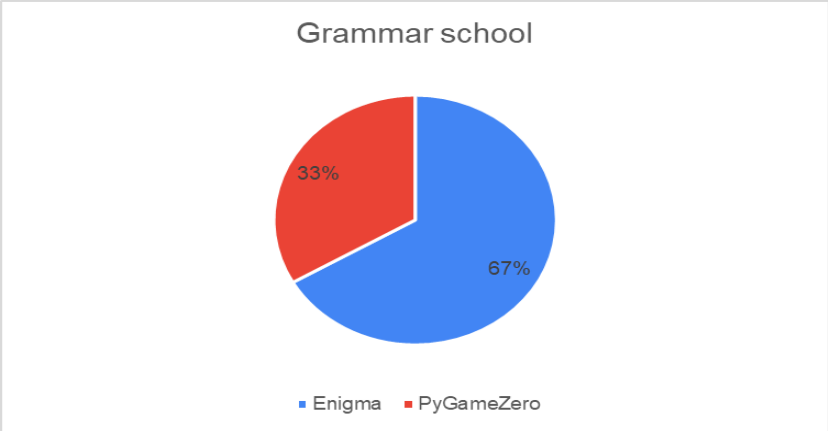


Figure 12. Ratio of participants from grammar schools on workshops

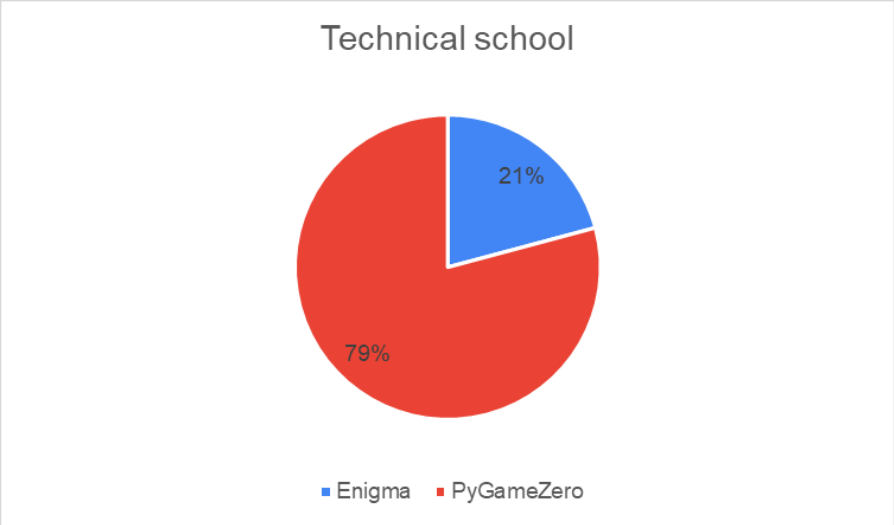


Figure 13. Ratio of participants from technical schools on workshops

6.3. Feedback analysis of Winter school of programming

Winter school of programming [2] was realized in 2022. Workshop was realized in Faculty of Management Science and Informatics, in duration of 3 days. Goal was to create a game using Java programming language and Greenfoot IDE. Attendees of the workshop had an opportunity to get a certificate of successful completion of the workshop, which granted the acceptance for the university studies. For the workshops, students of last two years in high schools were approved to enroll. Participants came from grammar schools, technical schools and vocational schools.

Mentioned workshop was led by the team of scientists and experts from the Faculty of Managements Science and Informatics. Object first principle as well as game-based development have been used. We collected feedback after the realization of the workshop which included following data: type of school, lecturers' opinion on what attendees learned, students' opinion on the lecturers and the workshop content.

As in the previous workshops, the general feedback was a positive one, both from the students and the lecturers. Participants from all three types were rated with regard to how much they learned during this workshop, and as can bee seen from provided graphs, a majority of them learned "A lot", while the rest of them learned at least "Something". Most importantly, no participants regardless of their school and prior knowledge were left without learning anything. Again, results indicate the validity of the chosen approach. Therefore, we consider educational materials for teachers on the base of game-based development and Greenfoot IDE to be a legitimate approach with potential to succeed in pedagogical praxis after pilot deployment.

[2] V. Lendel, „ONLINE škola programovania 2022 - registrácia pre stredoškolákov spustená," 16 12 2021. [Online]. Available: https://www.fri.uniza.sk/aktuality/online-skola-programovania-2022-registracia-pre-stredoskolakov-spustena.

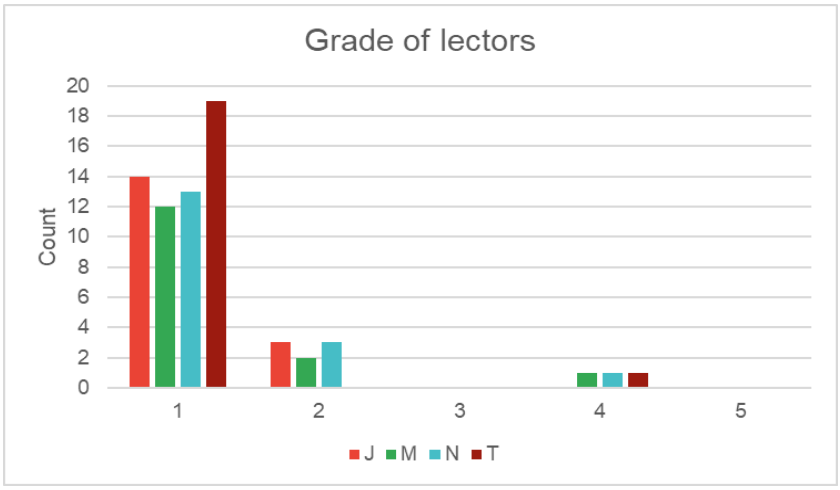


Figure 14. Personal opinion about the lector

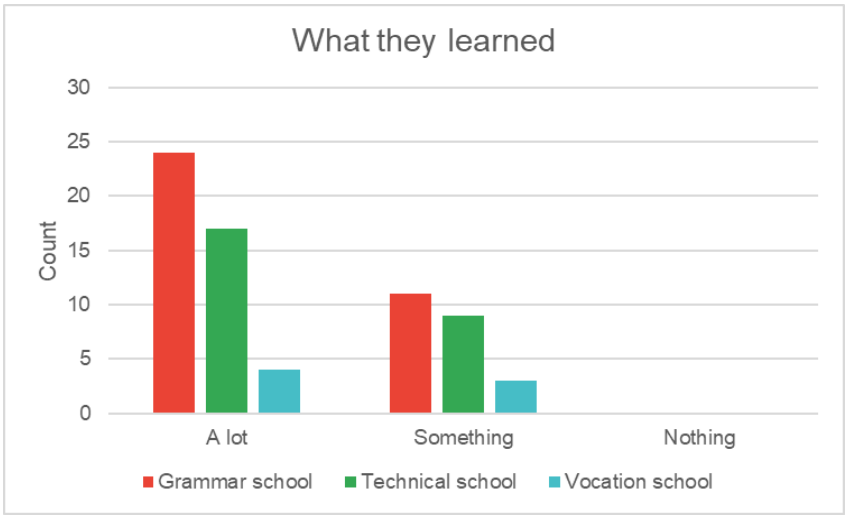


Figure 15. Personal opinion about the amount of new information students learned

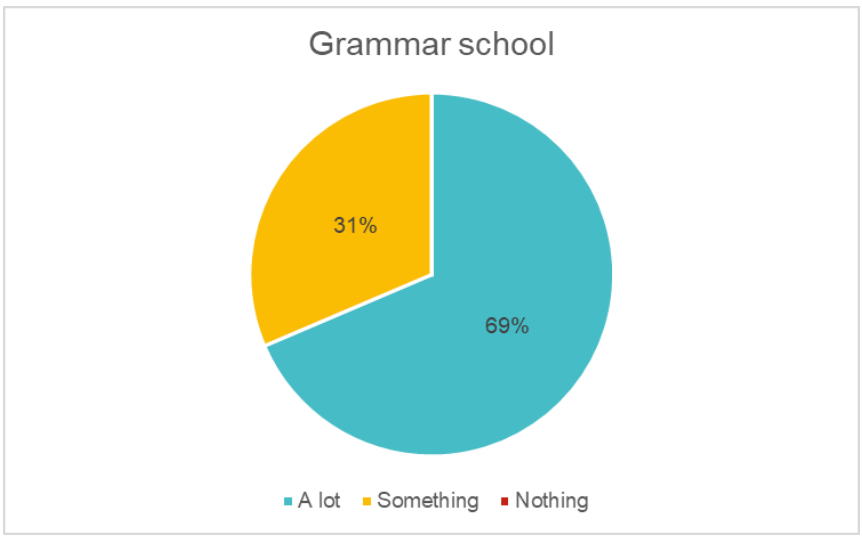


Figure 16. Ratio of personal opinion about the amount of new information students of grammar schools learned

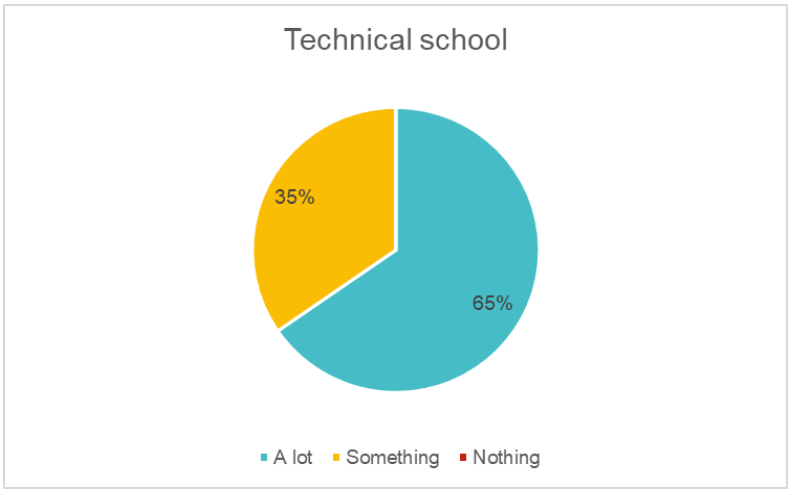


Figure 17. Ratio of personal opinion about the amount of new information students of technical schools learned

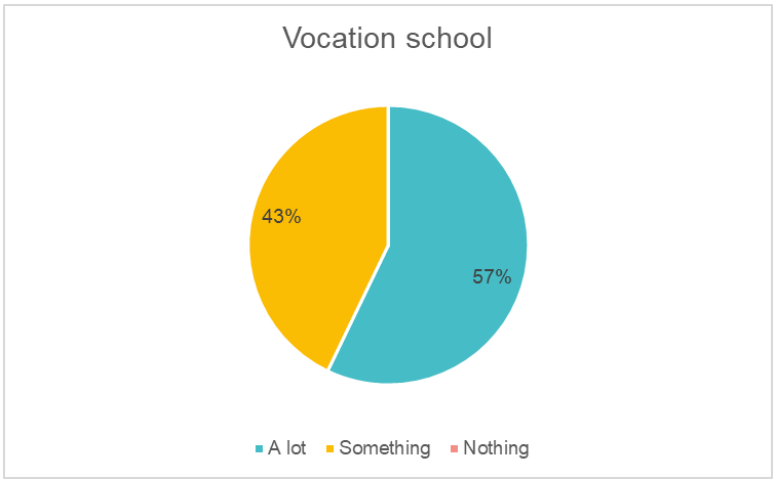


Figure 18. Ratio of personal opinion about the amount of new information students of vocation schools learned

6.4. Conclusions on empirical results from UNIZA

Three separate workshops with participants from different high schools and different years of study were conducted in order to investigate whether using a combination of interesting topics, OOP and game development has sufficient motivational potential to create an interest in programming and STEM fields. The gathered experiences and feedback confirmed this as a legitimate approach. Indeed, both students and lecturers gave positive feedback in terms of motivational and interesting topics, used technologies and the acquired skills. All three workshops shared interesting topics and game-based development, however, the programming languages and environments varied. This included using Python and Java with conventional IDEs, as well as Java with Greenfoot IDE. While workshop participants managed to successfully accomplish their tasks regardless of the chosen programming language, the Greenfoot IDE did stand out as it was particularly constructed for gradual and visual learning of OOP and game development.

7. Empirical results from GYPCE

Josef Rak <peparak@gmail.com>

7.1. Introduction

In the school year 2022/23 a test of usage of Greenfoot at Gymnazium Pardubice (GYPCE) was made. Students in the second and third year of study are chosen. Algorithmization is taught in the second year of study in mandatory subject *Informatics* and in the third year of study in optional subject *Seminar of programming*. Before this test only Scratch was used in this school to teach programming skills and only with small time dotation. Time allocation in the second year of study was 4 hours. In the third year of study 18 hours. In the second year of study Greenfoot used to show students how to make a simple computer game. After introduction to Greenfoot and showing basics of Object-oriented-programming concepts students received a pre-prepared project of a computer game to finish. In the third year of study, the game was made from the beginning. Due to greater hour allocation, 3 computer games were created. Two games by following the instructions from the teacher and one as an exercise on students' own.

The aim of the test was, if Greenfoot is suitable for teaching programming skills in the school and if students will be able to write code in Java. Idea in the school was to find a bridge between Scratch and Java, which is taught in the *Seminar of programming*.

Next motivation for using Greenfoot is to improve OOP skills and for better understanding of OOP principles.

7.2. Greenfoot test results

After the Greenfoot usage students were asked to fill the following form.

1. Was Greenfoot your first programming experience (Y/N)?
2. If not, write your programming experience (Scratch, Java, php, etc.).
3. I understood the Greenfoot environment and would be able to create a computer game in it (Y/N) ?
4. If so, what amount of time would you need in the timetable for this.
    a. Of that teaching hours
    b. Of that hours of independent work
5. I was intrigued by the Greenfoot environment: 1 worst – 5 best
6. I learned something new: 1 worst – 5 best
7. I take the experience of the Greenfoot environment as a benefit for my further education: 1 worst – 5 best
8. How user friendly is the Greenfoot environment: 1 worst – 5 best
9. How is Greenfoot applicable to teaching at our school: 1 worst – 5 best
10. I would be in favor of including the Greenfoot environment in the teaching of algorithmization at our school: 1 worst – 5 best
11. Describe in a few sentences what interested you in the Greenfoot environment, how you worked with the environment and how you liked Greenfoot.

12. What did you dislike about Greenfoot, what did you find difficult and what did you dislike?

Overall, 56 answers were collected from 8 students from the third year of study in optional subject seminar from programming and from 48 students from the second year of study in mandatory subject informatics. For the first question if Greenfoot was first programming experience the following results are obtained.

Figure 19. Greenfoot as first programming experience

For more detail of this result, the following graph and table shows student's programming experiences.

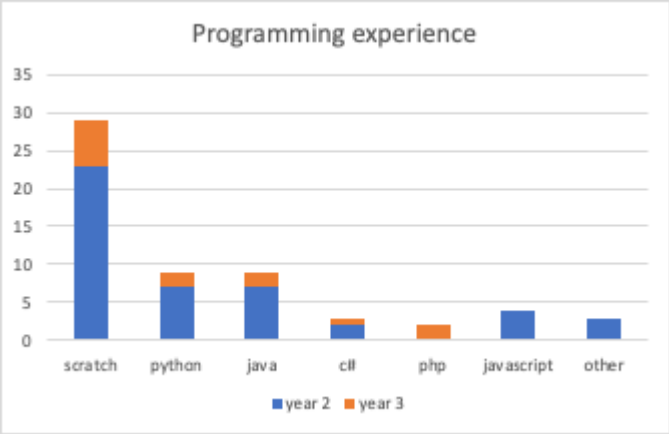


Figure 20. Experience in programming languages

Table 6. Experience in programming languages

	scratch	python	java	c#	php	javascript	other
2nd year	23	7	7	2	0	4	3
3rd year	6	2	2	1	2	0	0

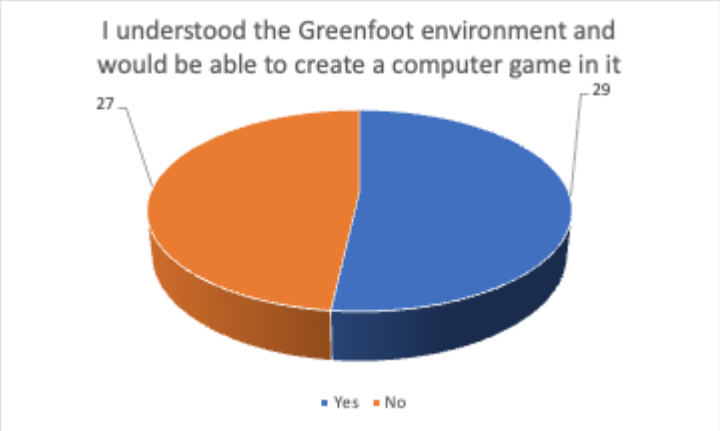


Figure 21. Understanding of Greenfoot environment

For the second question we recognized that more than 50 percent of students understand the Greenfoot environment and that they will be able to use Greenfoot to make their own computer game. Following graph shows how much time of teaching and independent work students think they need to create their own game. The average value is 3 hours of teaching and 3 hours of independent work.

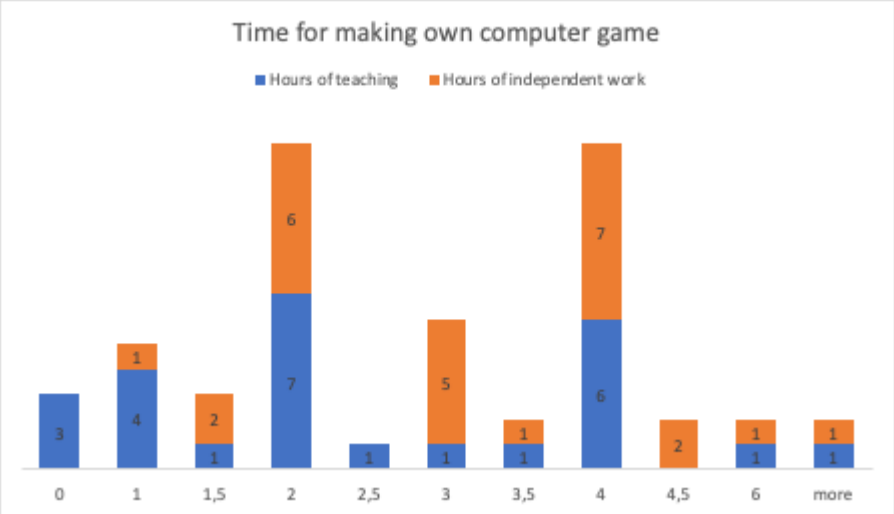


Figure 22. Time for making own computer game

For the last six questions (5 – 10), students were given the opportunity to answer on a scale of 1 – 5, where 1 meant the strictly no, 2 meant rather no, 3 meant neutral answer, 4 meant rather yes and 5 meant strictly yes.

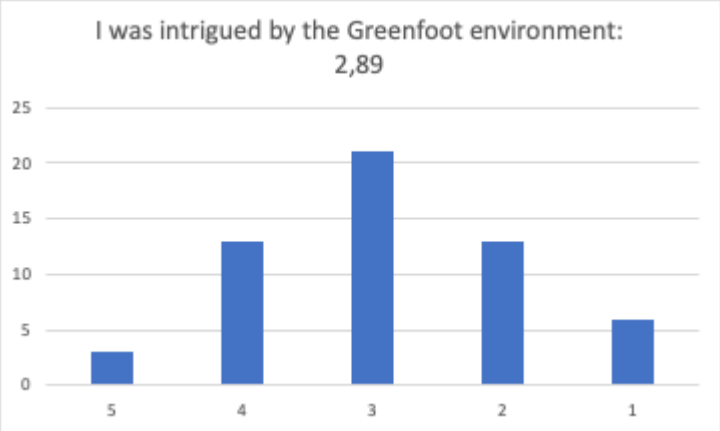


Figure 23. Intrigation with Greenfoot environment

Figure 24. New knowledge

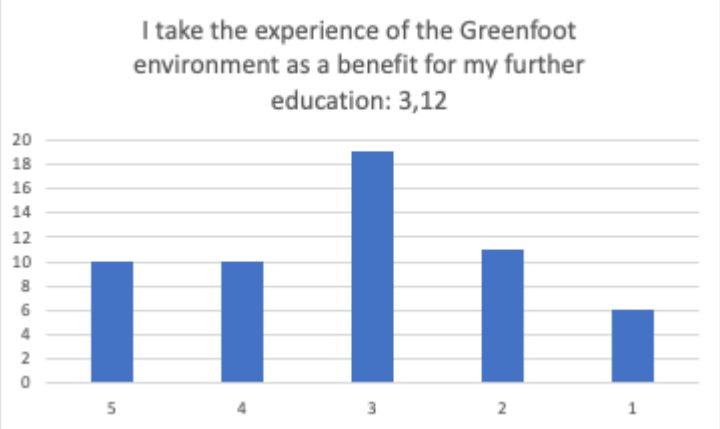


Figure 25. Benefit of Greenfoot

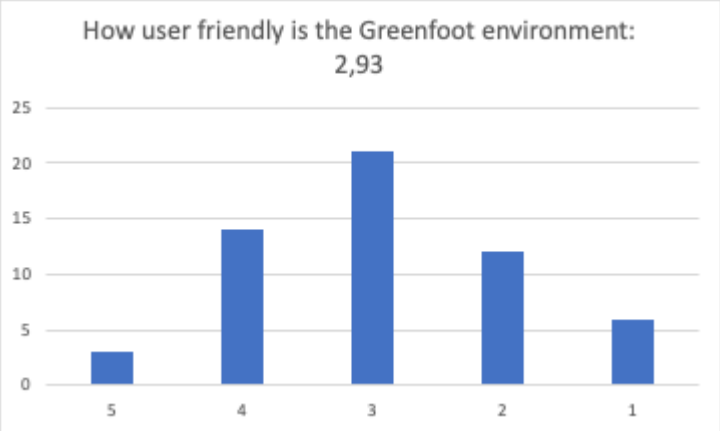


Figure 26. Friendliness of Greenfoot environment

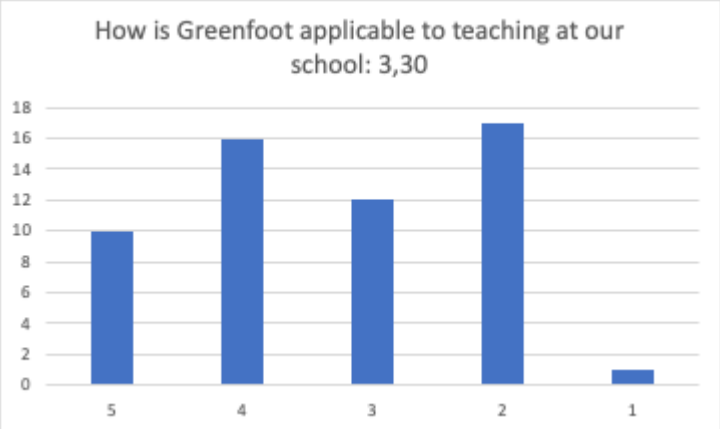


Figure 27. Greenfoot applicability in school

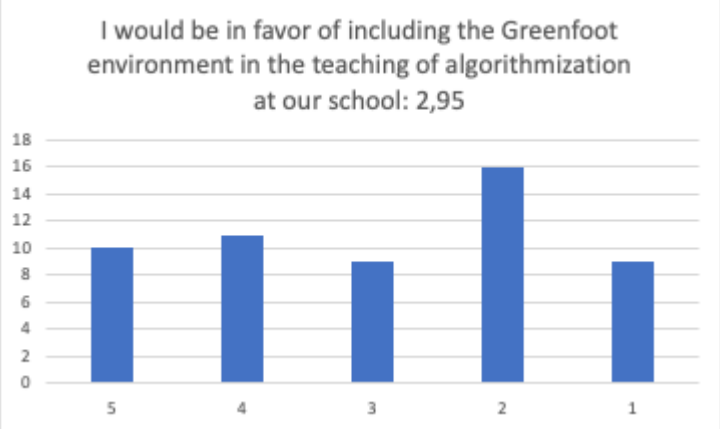


Figure 28. Favoritization of Greenfoot

From the graphs we can see that **feedback was positive**. It is important to emphasize that our school has a comprehensive study program and, in addition to students who are thinking about technical education, students of humanities, languages and medicine also took the test.

At the end let us write some student answers to the last two questions.

Q11. Describe in a few sentences what interested you in the Greenfoot environment, how you worked with the environment and how you liked Greenfoot.

- "quite interactive and fun, a bit of a problem to remember what to do and how to do it"
- „It is a different and probably more understandable way for many to learn and understand the programming process in an environment similar to java."
- "I was intrigued by the variety of options in greenfoot. After a few hours of instruction, it is really possible to create a relatively fun and sophisticated game."
- "I liked and enjoyed working in Greenfoot and its environment. It was a nice enrichment of the lesson and also a demonstration of a specific use of programming."
- "A good transition between simple programming (e.g. in Scratch) and programming in a more complex language. Comprehensible environment, well-designed graphics."
- "Transforming something almost abstract into something more imaginable and more graspable"
- "I worked well in the Greenfoot environment, the best thing, in my opinion, was that we saw for ourselves exactly what we were going to do by modifying the code, and therefore could better imagine what exactly would be done for the next code writing."

Q12. What did you dislike about Greenfoot, what did you find difficult and what did you dislike?

- "The overall environment, in my opinion, is not beginner friendly at all"
- "The complexity of the names when I want to use a function."
- "I didn't have enough time to understand the syntax."
- "Difficult navigation in the environment. Unsightly environment."
- "I would need more independent work, I didn't have enough time to explain it like this"

7.3. Conclusions on empirical results from GYPCE

Most important fact is that we recognized that more than half students are able to make simple computer games after a short Greenfoot course. We recognized that students were able to write Java code. From the feedback we understood that there were some problems with mistakes in Java code which is not intuitive for beginners. But there is possibility to start Stride and then continue with Java and also to give bigger hour dotation as was recognized from the feedback to question 4.

9. Innovative teaching and learning ideas

lmasnec@foi.hr, zstapic@foi.hr, ghajdin@foi.hr, dplantak@foi.hr

9.1. Teaching and learning approaches

Taking into consideration the results of all above mentioned activities we identified the following teaching and learning approaches that could be used in the classroom to increase the level of innovation and to increase the motivation of students to the programming in general.

- **Peer learning** - is a part of the learning process in which students learn from and with each other. It can be a standalone activity, such as a part of working in pairs or groups, or it can be a part of more complex learning activities such as project based learning.
- **Team teaching** - is a process of teaching where teachers work (most usually) in pairs in the same classroom. With this teaching approach teachers usually support interdisciplinary teaching and learning, thus facilitating holistic understanding of the topic at hand. Team teaching can be conducted with a pair of teachers of the same course, especially in larger classroom groups when teaching complex topics which require a lot of individual student-teacher interaction.
- **Inquiry learning** - students can explore different questions which arise during the learning process. Inquiry learning can facilitate student's process of discovery, especially if they are seeking for new ideas or possible solutions. It is also an integral part of more complex teaching and learning approaches, such as project based learning.
- **Flipped classroom** - teachers can facilitate students in the learning process through guidance and instructions related to acquisition of initial basic knowledge at home, as a part of homework, usually accompanied by video teaching materials, which helps students prepare for the class. In this approach understanding and application is jump started with initial knowledge and questions , thus during the class students can focus on higher levels of knowledge and receive support from a teacher in school.
- **Problem-based learning** - problem-based learning can be based on the hypothetical, but also real-world problems which boost students' motivation. One of the integral parts of problem based learning is problem posing where students need to identify the problem, deduct what they know and what they must solve. To solve a problem students usually need to acquire new knowledge or think of creative ways of applying existing knowledge in a completely new scenario.
- **Interdisciplinary learning** - real life situations are often complex and utilize knowledge from different subjects. Interdisciplinary learning facilitates learning through a combination of different school subjects. It can be further supported through team teaching, project based learning or other complex learning and teaching approaches.
- **Blended learning** - is an educational method that combines traditional classroom learning with online instruction.
- **Gamification** - is based on implementation of game mechanics in the teaching and learning process. The goal is to positively influence students' motivation and progress. . For example, for each correct answer that a student gives to a given question , he would receive 100 EXP (experience points). These experience points can later be used for bonus points or "unlocking" a new advanced topics that provides more information on a given subject.
- **Visual learning** - emphasizes the use of visual aids, such as diagrams, charts, video and animations, to enhance the learning process. In the context of teaching

programming through web applications, multimedia learning can involve interactive visualizations that help students understand abstract concepts more easily.

- **Block-based programming** - is programming where code is represented as blocks or puzzle pieces that can be dragged and dropped to create visual programs. This is for beginners and easy to understand. In a web application setting, block-based programming interfaces often provide immediate visual feedback, allowing students to see the results of their code changes instantly.

9.2. Teaching and learning materials related to object-oriented programming (OOP)

Taking into consideration the results of all above mentioned activities we identified the following teaching and learning materials related to object-oriented programming (OOP) that could be used in the classroom to increase the level of innovation and to increase the motivation of students to the programming in general.

- **Alice Programming** - offers a lot in the context of learning object oriented programming. In particular, setting up a "computer game"/animation scene involves clarifying the context of the procedure and the class of objects, and visually shows the execution of the procedure on the objects. The form of the course can certainly be used. Thus, tasks with attached files of the beginning/end of solving and videos that deal with these tasks.
- **Greenfoot teaches OOP with Java** - create 'actors' which live in 'worlds' to build games, simulations, and other graphical programs. Greenfoot is visual and interactive. Visualization and interaction tools are built into the environment. The actors are programmed in standard textual Java code, providing a combination of programming experience in a traditional text-based language with visual execution. Software for learning programming in a simple and fun way. There is a book: Introduction to Programming with Greenfoot, Object-Oriented Programming in Java with Games and Simulations. On the web is the Greenroom - a teacher community and provides resources (slides, worksheets, project ideas, tests, etc.) and a teacher discussion forum. You need to register and log it to look at the community because the OOP4Fun project application mentions Greenfoot.
- **Computer thinking and programming** - material for students to independently acquire content related to learning outcomes: define a logical expression for a given problem; analyze the problem, define input and output values and identify steps to solve the problem; apply simple data types and argue their selection, apply different types of expressions, operations, relations and standard functions for modeling a simple problem in the chosen programming language. The material consists of text, images, videos and interactive objects.
- **Virtual classroom** - group work in a virtual classroom (Teams). For example, in our use-case, students could be divided into groups that represent classes. One student defines attributes for a specific class, another one a method, another student another method etc., and finally they're joined into groups by classes they defined.
- **Textual online course** - introductory C++ tutorial designed to give the user understanding of how this language works.
- **Data discovery** - the lessons are designed to engage students with real-world data relevant to content taught in middle school and high school science courses. The Python lessons guide students in computational thinking to create simple programs to manipulate data. The lessons also provide students (and teachers) with instructions and guidance in the use of these technologies. Worksheets and supporting files are linked to from links at the top of each lesson webpage and from the downloads page.

10. Aligning results with PR1 results

In the following chapter we have tried to align the results of our analysis with the gaps and needs recognized in the results of the vertical analysis performed within project result 1 (PR1). The table below enumerates mentioned gaps and needs on the left hand side and on the right hand side brings possible novel and innovative teaching approaches and ideas which could be used to fulfill them.

10.1. PR1 and PR2 results allignment

The results are presented from the point of view of two analyzed perspectives:

- Perspective on curriculum development, learning outcomes, teaching materials and teaching activities
- Students' and teachers' point of view regarding teachers' competences

Table 7. Mapping gaps to possible teaching approaches

PR1 findings	PR2 findings
Perspective on curriculum development, learning outcomes, teaching materials and teaching activities	
Teachers recognized that a new curriculum should be developed.	In high schools, OOP should be introduced by topics covering basic programming concepts in the beginning and narrower topics related to OOP would be more appropriate in separate courses. It is crucial to connect and encourage information exchange between school and university teachers, with the involvement of policymakers who define curricula related to programming skills at all educational levels. From a course/instructor perspective and from a course design perspective, the following innovative forms of instruction/knowledge transfer should be used: Blended learning, learning-by-doing, problem solving, collaborative problem, teamwork, problem-based learning, active learning, lab-based learning. In addition, various forms of innovative approaches should be applied in lectures, seminars, and laboratory exercises.
The curricula of IT subjects in high schools shows that OOP is poorly represented, with an insufficient number of teaching hours.	
High school curriculum does not recognize the importance of the use of innovative tools (such as Alice) which could help in overcoming lack of motivation and knowledge shortcomings in structural programming and OOP.	

	Games and gamification in general were frequently used to motivate students to program. Students liked the opportunity to be creative or to compete for knowledge when supported by an appropriate setting. From the results of the literature review, three different types of learning through games were suggested: learning by playing, learning by creating games, learning by using game-related tools and learning with gamification.
Most university study programs teach programming from the beginning, because of the lack of a homogenous prior education of students.	The overall goal of PR2 was to find appropriate and innovative learning and teaching ideas and approaches that would solve these issues. As mentioned in the previous chapters there are several identified good practices that could be used to improve the achievement of learning outcomes during the highschool education. However, it should be noted as well that curriculum redesign should result in introduction of OOP topics and set goals in achieving OOP related learning outcomes as well. It is also important to select an appropriate type of assessment: (online) questionnaires are the only accepted method for assessing students' enjoyment, usefulness, interest, engagement, and simplification of programming and OOP concepts which will be defined on the high-school and not on university level.
Freshmen students in general don't possess any notable competencies or skills related to programming acquired at the high school level.	
Student-to-student experience shows that most of the students struggle with the very basics of programming concepts.	By using teamwork in OOP assignments, students would have the opportunity to share their knowledge and transfer the implementation of basic programming concepts to other students (peer-to-peer learning).
Students with higher prior knowledge reported that it was acquired by self-learning activities which shows that highschool students are	This project's results will yield a set of materials which will give a chance to highly motivated students with solid prior-knowledge to increase that knowledge through different

capable of self-learning algorithmic problem solving if provided with good materials.	activities and roles. Such students could significantly improve the overall achievements of the whole group if they will be given a chance to share the knowledge or to lead the teams.
For the first few weeks or months students seem not to know “where they are and what is going on”, and what is even worse they seem not to be very interested in programming and they see no purpose in programming knowledge.	As several PR2 outputs point out, the main goal should be to incorporate learning and teaching tasks into stimulating and entertaining activities that will have a positive impact on higher attendance and completion rates. This would increase the interest of high school students for programming in general and eventually lead to better understanding of programming and OOP concepts. In that case students would not be “lost” when faced with university curricula.
Education in the field of information technologies is usually associated with colleges and higher schools, but practice has shown that the training of future IT professionals should start much earlier.	Use of more different tools and programming languages in earlier years of study should be encouraged. Programming concepts could be simplified using visualization tools. Although some countries have already introduced the use of some tools like Logo or Scratch, these are interesting to elementary schools but not to high schools. Thus, more advanced tools that are designed to support OOP should be used. We have recognized that Alice and Greenfoot stand out among other tools.
The starting point and basis for effective improvement of programming learning are high-quality learning materials supported by teacher training, classroom work, and students' independent work activities.	As students reported, for teaching programming it is important that teachers use novel and up-to-date teaching materials and employ creative teaching methods. Also, the teacher's availability and flexibility to work with students outside the classroom is necessary to motivate students and generate greater interest in the subject. Innovative practices used in lectures that were mentioned are the demonstrations by examples, teamwork, discussions, short quizzes, usage of multimedia and interactive materials when explaining and executing the code, invited lectures or other display of real life examples.

<table>
<tr><td></td><td>There weren't many mentions of innovative practices used in exercises, but those mentioned were teamwork, recorded materials for learning at home, and demonstrations by teachers followed by students' further research.</td></tr>
<tr><td>High schools should "open their doors" to a new generation of future computer professionals, the generations that grew up with Facebook, Google and other Internet services.</td><td>For learning and teaching of OOP concepts, learning by creating the games showed the significant effects for improving students problem solving skills and engaging them in a fun and entertaining environment.</td></tr>
<tr><td colspan="2">Students' and teachers' point of view regarding teachers' competences</td></tr>
<tr><td>Teachers recognized a need for a new legislative framework to cover issues related to the lack of motivation for teachers with programming knowledge to come and teach in high schools, as, in their opinion, motivation of high school teachers is very important.</td><td rowspan="4">The lack of a legislative framework which would motivate teachers to embrace novel and innovative teaching and learning methods in programming and in OOP programming plays a significant role. Thus, national strategies and development agendas should introduce a set of employment and motivational measures that would solve this issue on the implementation level. Although it is not in the scope of this project to deal with these issues, we strongly believe that the project results including novel teaching and learning materials and approaches would increase the teachers' motivation for this topic.

From student's experiences it was not possible to determine what skills teachers lacked, but the results clearly indicate what skills teachers should have regardless of their prior educational background. Nonetheless, knowledge of the subject matter and how well teachers are able to explain the material (e.g., using their own examples) are the critical skills teachers should possess.

Other skills recognized as important are:
• Communication skills - how do teachers communicate with students (e.g. include them in discussions)
• Presentation skills - how well do teachers explain the materials in their lectures</td></tr>
<tr><td>IT experts are not eager to teach in high schools, where salaries are low, so the principals are forced to employ teachers who have only partial knowledge of informatics, and they are unable to cover higher demands in teaching programming.</td></tr>
<tr><td>In Serbia and Croatia, as well as in Slovakia and Czech Republic there is a significant problem with knowledgeable staff teaching computer science and mathematics.</td></tr>
<tr><td>There is an issue that final-year students in Serbia and Croatia are encouraged to apply as teachers in the high schools. In general, there are not many professors who are educated, educated to teach students the basics of programming. Usually teachers who do not have a full fund of classes in their primary subjects such as Mathematics are teaching informatics as well.</td></tr>
</table>

	• The use of real-life examples - how well do teachers connect real life examples with the lectures they're holding • Pedagogical and methodical skills - developing and enhancing teachers' pedagogical content knowledge and ability to apply effective instructional methods and tools to help students to overcome problems with programming concepts in OOP. Students mostly noted that teachers should motivate students by explaining why a certain topic is being lectured, by giving interesting real-world stories related to the materials, by being available for additional explanations, etc. Teachers should organize and structure learning materials well and not get lost in them.

10.2. Final remarks on the alignment

The overall conclusion from PR1 findings is that "*there is a lot of work in front of all stakeholders, legislation enforcers to prepare stable and motivating infrastructure and environment, high-school curriculum designers to take into consideration the growing need for programming knowledge and STEM in general, institutions educating teachers of informatics to enable them to teach programming and related concepts, university curriculum designers and university teachers to build on high-school knowledge and to students to take every opportunity to acquire the skills and competencies required in the future dynamic market*".

Blended learning and flipped classrooms were recognized as the most dominant designs used to improve students' programming skill and motivation. The importance of fostering learning of core principles of programming by focusing on student learning rather than instructor teaching was emphasized. Various authors emphasized the importance of establishing the context and framework that will enable students to deconstruct and reason about, as well as the importance of providing ample scope for individual critical thought, design, and creativity. The integration of multiple languages was also encouraged. Furthermore, flipped classroom teaching approach was suggested as most suitable for active learning/teaching.

In general, it is crucial to connect and encourage information exchange between school and university teachers, with the involvement of policymakers who define curricula related to programming skills at all educational levels. This unified approach would make high school students more ready and skillful for entry level at university programmes containing programming courses

Developing and enhancing teachers' pedagogical content knowledge and ability to apply effective instructional methods and tools to help students to overcome problems with programming concepts in OOP.

Following the repository analysis and literature review, the use of **Greenfoot** should be introduced in introductory OOP courses and supporting materials should be based on gamification. This approach will, in the experience of faculty from PR1, provide a solid foundation for students to become more adept at learning algorithmic problem solving independently. The benefits would be twofold: (1) low proficiency/no proficiency students would be able to develop basic skills and competencies related to OOP programming, and (2) students who already have some prior knowledge could be further engaged in active instruction, which would encourage them to actively participate in class. This could reduce dropout rates in the courses and increase students' motivation from the beginning.

11. OOP4Fun Learning Design

11.1. Learning design methodology

In order to make more informed decisions on how they approach designing learning activities and interventions, a learning design methodology (LD) is used. (*IO2. Learning Design Models*, 2020) It enables teachers to be more informed and to be pedagogically informed. LD is a well established approach to designing learning activities and in recent years there have been several strong research initiatives that are trying to deepen the understanding of how LD influences and how it is influenced by new technologies and innovative pedagogical practices.

One example is the Integrated Learning Design Environment (ILDE), which is a networked system integrating collaboration functions, design editors and middleware that enables deployment of the designed learning situations into Virtual Learning Environments. Another example is CompendiumLD, which uses a flexible visual interface that enables practitioners to articulate their ideas and map out the learning design as a set of learning outcomes (*An Integrated Environment for Learning Design*, 2018).

Some other examples of well-implemented Learning Design tools include ScenEdit, IAMEL (Bottino et al., 2010), and LDTool (Agostinho, 2011). These tools aim to support practitioners in their task of creating more innovative and effective computer-supported learning situations. Bottino says that the teacher's primary role shifts from that of information giver, to that of facilitator and guide, someone who has to incorporate mediation, modeling, and Coaching. Bottino claims that this requires a high degree of adaptability to new schemes, models and tools on learning and teaching. When it comes down to managing technology, that may take up a great deal of time and intellectual energy. She resumes by stating that in the new educational landscape, teachers and all those involved in designing and enacting learning processes (trainers, pedagogical experts, designers, researchers etc...) are increasingly required to take into account a huge variety of different elements, in an effort to ensure that these form part of a coherent, manageable whole that responds effectively to learners' needs and that consents the full attainment of the intended educational objectives.

Learning design methodology is a process for creating effective and engaging learning experiences for students. It involves the systematic analysis of learning goals, the selection of appropriate learning activities and resources, and the evaluation of learning outcomes. The process typically begins with an analysis of the learning goals and objectives, followed by the identification of the target audience and their learning needs (Sharpe et. al., 2010). Conole and Fill (2005) define a set of points how to approach developing a learning design toolkit:

1. Work closely with practitioners to analyze their methods, when creating or re-purposing resources, and be guided by their requirements.
2. Enshrine good practice within the toolkit, such that it will guide and support teachers as they create, modify, and share teaching and learning resources.
3. Research, understand and apply what is going on in the learning design field, particularly evolving standards in the areas of sharing digital resources, interoperability, searching, repurposing, and permissions.

4. Embrace new technologies, such as adaptive hypermedia and semantically structured metadata, to provide a tailored development environment, accessing heterogeneous data repositories across a grid service infrastructure.
5. Develop, test and evaluate a prototype toolkit with practitioners and then revise in light of feedback.

Same authors define three more purposes for the learning design toolkit :

1. As step-by-step guidance to help practitioners make theoretically informed decisions about the development of learning activities and choice of appropriate tools and resources to undertake them.
2. As a database of existing learning activities and examples of good practice which can then be adapted and reused for different purposes.
3. As a mechanism for abstracting good practice and metamodels for e-learning

Conole et. al. (2008) emphasizes the internet's impact on education. They say: “We now have a wealth of research literature on the myriad of ways in which teachers have experimented with different technologies to support learning.“ Conole et. al. state that although technology has the potential to enhance learning, there exists a discrepancy between what technology can do and how it's actually used. This disparity is partly due to insufficient comprehension of how technology can be employed to provide specific learning benefits, as well as a deficiency of appropriate guidelines for designing effective technology-based learning experiences.

Conole et. al. report seven reasons why adopting a learning design approach can be beneficial (Conole et. al., 2008):

1. It can act as a means of eliciting designs from academics in a format that can be tested and reviewed
2. with developers, i.e. a common vocabulary and understanding of learning activities.
3. It provides a means by which designs can be reused, as opposed to just sharing content.
4. It can guide individuals through the process of creating new learning activities.
5. It facilitates reflection by the designer, by making the process more explicit
6. It creates an audit trail of academic design decisions.
7. It can highlight policy implications for staff development, resource allocation, quality, etc.
8. It aids learners in complex activities by guiding them through the activity sequence.

To conclude this chapter, a learning design is a visual depiction of the educational journey you've created for your students, outlining the order of learning activities they'll participate in, and the materials and assistance available to assist them with each task. A learning design methodology specifies how to be an architect of such a journey.

11.2. Innovative teaching scenarios for OOP4Fun

As mentioned in the introductory chapters and in the context of education and instructional design, teaching scenarios (TSs) represent detailed descriptions or narratives that outline a specific instructional situation or context. These scenarios are often used in teacher training

to simulate real-world teaching situations and thus we find it as the best tool to represent our innovative teaching and learning ideas. As teaching scenarios typically include information about the learning objectives, the content to be taught, the characteristics of the learners, the instructional methods employed, and the assessment strategies used, it can be aligned with the elements needed for our learning design artifacts as well.

In order to have a structured approach in defining several teaching scenarios we have defined the following template that would be filled with concrete data related to a specific learning scenario. Template contains a short description on how to define each element of the TS.

Table 8. Template for documenting learning scenarios

Title	Give learning scenarios a descriptive and an attention-grabbing title.
Learning objectives	Clearly state the intended learning outcomes. What should students know, understand, or be able to do by the end of the scenario?
Target audience	Specify the intended audience, grade level and pre-existing knowledge for which the learning scenario is designed.
Scenario duration	Estimate the time required to complete the learning scenario, including any specific timeframes for different activities. E.g. teacher introduction (5 min), research done by students individually (10 min), programming solution in team (20 min), presenting/discussion (10 min).
Materials&resources	List the materials, resources, and tools needed for both teachers and students. This could include textbooks, online and multimedia resources, software, etc.
Description	• Introduce the learning scenario, explain its purpose and relevance. • Outline the core activities that students will engage in to achieve the learning objectives. Include details such as discussions, hands-on activities, group work, competition, etc. • Specify how students will be organized, i.e. are they going to work individually or in a team. How large are teams gonna be? • Explain what projects/problems/tasks will students be working on. Recommendation is to use problem-based or project-based approaches. Also, these should reflect real-life situations.

	• Explain how projects/problems/tasks will be assigned to students (teams). • If working in teams, provide details on how they are going to collaborate. • Provide more details related to activities that students need to participate in. • If e.g. flipped classrooms are used specify what part of the given topic students need to research by themselves.
Assessment	Provide details related to how students' effort and knowledge will be assessed. • Who is going to assess the students: (1) teachers, (2) students their own work (self-assessment), (3) students each other's work (peer-assessment) • What evaluation criteria will be used? • How often will assessment be performed? • etc.
Result dissemination	Explain how students are going to disseminate their results to teachers and fellow students. E.g. students (all or subset) can present their results/solutions in front of the teacher and their peers, and then the comparison and discussion may follow.

11.2.1. TS1: Introduction to Greenfoot: Exploring Game Development with Creativity

Title	Introduction to Greenfoot: Exploring Game Development with Creativity
Learning objectives	By the end of this session, students will not only have successfully installed Greenfoot and witnessed its capabilities through example projects but will also have engaged in collaborative, hands-on play within the development environment. This playful introduction sets the tone for an exciting exploration of game development, encouraging creativity, teamwork, and an enthusiastic approach to coding with Greenfoot.

Target audience	Secondary school students attending the OOP4Fun course. Basic programming knowledge including variables, functions, iteration and selection concepts.
Scenario duration	Introduction (5 minutes) Rush-hour challenge (10 minutes) Playing games with teacher (30 minutes) Team Formation and Project Assignment (5 minutes) Team Collaboration and Coding (30 minutes) Peer Review and Feedback (10 minutes) Homework (30 min) Competition grading (30 min)
Materials & resources	Greenfoot webpage and download instructions. Examples prepared by the teacher. Internet resources for identification of other examples.
Description	In this 90-minute learning scenario, secondary school students will dive into the Greenfoot world by means of gamification, fun, research and teamwork. After the teacher **introduces** today's session, reflects on the previous session and sets challenging goals, the **rush-hour challenge** begins. Students are given the gamified assignment to find instructions, download and install greenfoot (yet unknown development tool for them) on their computers. The first three students are given tokens of appreciation (badges, points, scores, sweets etc.). The second surprise for them is that in the next 30 minutes they will be **playing games with the teacher**. This is a teacher guided session on opening, compiling and running one-two simple example projects (on the introductory to medium level of complexity). This will show students the basic elements of the Greenfoot development environment as well as of basic procedures of handling the project files and assets.

	Afterwards, the students will be **grouped in the teams** (3-4 students each) and will be given a simple assignment. Teams should change "something" in the given example project to make the game surprising or fun. Team **collaboration and coding** (30 minutes) will have teams work collaboratively on trying to change something in the given examples. If they break the code beyond the line of being able to fix it on their own they can ask for help from the teacher or can download the "start version" again. This will be a good example why we should use version control systems when coding. One or two teams will present their work for **peer review and feedback** and the group will discuss the results along with the teacher. At home for **homework**, each student should search for examples of Greenfoot games and should introduce his class to his favorite example by uploading a link, description of what makes it his favorite example and two-three screenshots of the development environment and running game. As part of the **gamification** and motivation via **competition**, each student should vote for three best games (it is not allowed to vote for his own game). The winners are announced and awarded with tokens of appreciation (badges, points, scores, sweets etc.).
Assessment	This activity will enable teachers to give formative assessment feedback based upon the discussions and monitoring of students' flipped classroom and teamwork. The gamification represents non formal assessment but will increase the interest, intrinsic motivation and learning outputs of the whole group. The peer-review assessment will be performed online as a part of a homework assignment. This will remind students of important aspects of the session, will make them install Greenfoot, open different examples and remind them of what was done during the class which will increase the overall achievement of learning outcomes.
Result dissemination	In order to disseminate their results to teachers and fellow students the usual setup of the Learning management system (e.g. moodle will be used). The students can continue the discussion on the topic on the forum that is provided to them via the Learning management tool.

This learning design is also implemented in the learning design tool and can be found here (http://learning-design.eu/en/preview/bd6a3cf90d09a6b6495ecf82/details) and its delivery aspects are presented in the figure bellow:

Figure 29. TS1: Introduction to Greenfoot in learning design

11.2.2. TS2: Exploring Classes and Objects through Game Development with Greenfoot

Title	Exploring Classes and Objects through Game Development with Greenfoot
Learning objectives	The purpose of this learning scenario is to introduce secondary school students to the fundamental concepts of classes and objects in object-oriented programming (OOP) using the Greenfoot tool. By immersing students in a hands-on project to develop a simple game, this scenario aims to make the abstract concepts of OOP tangible and applicable in a real-world context.
Target audience	Secondary school students attending the OOP4Fun course. Basic programming knowledge including variables, functions, iteration and selection concepts. Students should be introduced to Greenfoot in general.
Scenario duration	Introduction (5 minutes): Teacher introduces today's session, reflects on the previous session and sets challenging goals 😳.

	Flipped Classroom Session (10 minutes): Students conduct independent research on what objects are to be presented on the stage of the game they are developing. Discussion (10 minutes): Teacher guided discussion on recognized objects and their classification in classes. Greenfoot tutorial (25): Teacher guided tutorial on creating selected objects in the game, emphasizing classes and objects. Team Formation and Project Assignment (5 minutes): Formation of teams (3-4 students each) and assignment of specific project that will build on the outputs of the previous activity. Team Collaboration and Coding (20 minutes): Teams work collaboratively on designing and implementing their assigned projects. Peer Review and Feedback (15 minutes): One or two teams will present their work and the group will discuss the results along with the teacher. Homework (30 min): Each team uploads their project, and peers provide constructive feedback via homework activity. Each student will evaluate one-two solutions from other teams. Project work (60 min): Each team will utilize the knowledge by defining game objects in the project they are working on continuously through the whole year.
Materials & resources	The textbook from the OOP4Fun project. Resources from OOP4Fun project. Project source code from Github/Gitlab. Internet resources.
Description	In this 90-minute learning scenario, secondary school students embark on a captivating exploration of object-oriented programming (OOP) principles of *classes and objects* through the lens of game development using the Greenfoot tool. The session commences with a concise 5-minute introduction by the teacher, establishing context, reflecting on prior learning, and presenting challenging goals for the day. The core activities encompass a 10-minute Flipped Classroom Session, during which students independently research and identify objects for their game. This is followed by a 10-minute teacher-guided discussion on the recognition and classification of these objects into classes. The heart of the session unfolds with a 25-minute Greenfoot tutorial, where students actively participate in creating selected objects, emphasizing the

	implementation of classes and objects. Team formation and project assignment take 5 minutes, with teams of 3-4 students each assigned specific tasks building upon the previous activity output. The subsequent 20-minute team collaboration and coding phase fosters teamwork, problem-solving, and practical application of OOP principles. The session concludes with a 15-minute peer review and feedback activity, where each team uploads their project for constructive feedback from peers as a homework assignment. This well-rounded learning scenario seamlessly blends individual research, class discussion, hands-on tutorial, collaborative coding, and peer evaluation, providing students with a holistic and practical understanding of OOP concepts in the context of game development. The students will have flipped-classroom research work, teacher-guided hands on activities and teamwork with pear review. The students will continue working on the game that was explained and started previous sessions. The overall approach of the game development will be a problem-based approach where students will each session be faced with a certain problem (e.g. what object we will have on some particular screen), and the end of the session will provide a solution to the given challenge as well as introduction of novel OOP concepts. All learned concepts will be implemented in the team-project that students are working on for their final grade.
Assessment	This activity will enable teachers to give formative assessment feedback based upon the discussions and monitoring of students' flipped classroom and teamwork. The peer-review assessment will be performed online as a part of a homework assignment. This will remind students of important aspects of the exercise, will make them critically assess other students' work, will give them insights into good or not so good solutions of their peers etc, and will increase the overall achievement of learning outcomes. The work in the team-project that the students are working on will also use these learning outcomes and knowledge.
Result dissemination	In order to disseminate their results to teachers and fellow students the usual setup of Github/Gitlab and Learning management system (e.g. moodle will be used). The students can continue the discussion on the

	topic on the forum that is provided to them via the Learning management tool.

This learning design is also implemented in the learning design tool and can be found here (http://learning-design.eu/en/preview/bd6a3cf90d09a6b6495ecf82/details) and its delivery aspects are presented in the figure bellow:

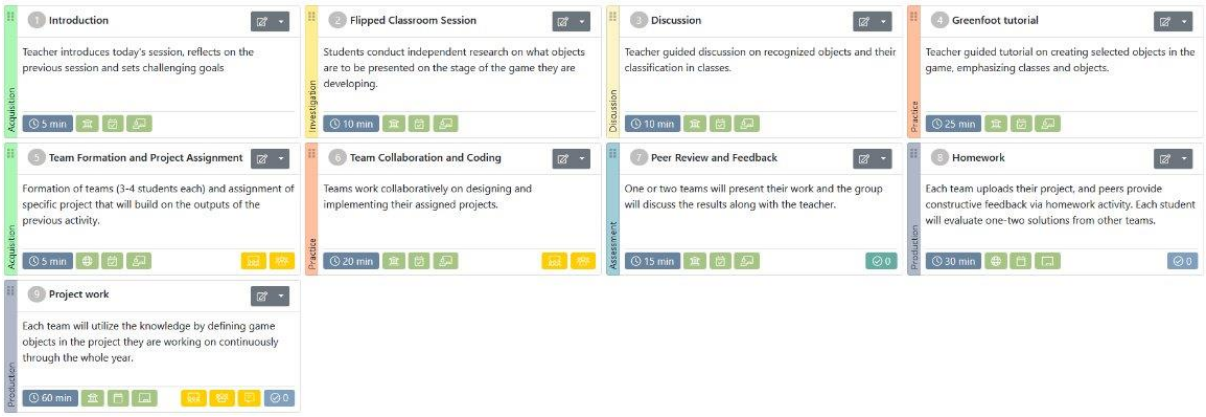


Figure 30. TS2: Exploring classes and objects in learning design

The usefulness of the visual representation in the tool are supplemented with the detailed analysis of the quality of the learning design in general with different analysis reports.

12. Conclusions

Through the course of this document, we presented our varied efforts aimed at identifying, describing and analyzing useful ideas for teaching object oriented programming. These efforts resulted in several activities starting with analyzing existing teaching materials repositories, performing systematic review of relevant research literature, then collecting university student experiences and opinions through a series of interviews, and finally reporting on the results of conducted programming workshops by one of the partner institutions.

The analysis of existing repositories of teaching materials included: MERLOT database, OER Commons database, and Croatian repository of open educational materials. The MERLOT analysis revealed a scarcity of teaching materials on object-oriented programming. A subsequent investigation of 44 materials from OER Commons and the Croatian repository identified 19 relevant resources for the PR2 goal of innovative teaching approaches. Deeper analysis focused on object-oriented programming topics and pedagogical practices, highlighting the scarcity of open educational resources specific to OOP. Some resources showcased platform-specific applications, while others integrated contemporary pedagogical methods. Lessons learned included the need for integrated solutions combining programming

and pedagogical aspects, a lack of emphasis on evaluation, and the suggestion to adopt problem-based and project-based learning for a more realistic approach to programming.

During systematic literature review, blended learning and flipped classrooms emerged as predominant designs for enhancing students' programming skills and motivation, emphasizing student-centered learning over instructor-centric teaching. The literature stressed the importance of contextual frameworks for deconstructing and reasoning about programming principles, fostering individual critical thought, design, and creativity. Integrating multiple languages was encouraged, with the flipped classroom approach deemed most suitable for active learning. Games were widely employed to motivate students, allowing creativity within a supportive framework, and collaborative activities were shown to enhance engagement, confidence, retention, and performance. Learning through games was explored in three ways: playing games, creating games, and utilizing game-related tools. Evaluation metrics included difficulty estimations of code-writing tasks, final examination grading, and students' passing ratios. Various tools, such as assessment systems, compilers, program submission systems, drill-and-practice tools, and intelligent tutoring systems, were developed to support programming education, effectively enhancing student learning.

Collected experiences and opinions from university students revealed that current high school programming education has little to no impact on future OOP knowledge at university. While this might be attributed partially to insufficient time dedicated to programming topics in high school, a large part of this problem can be explained by the challenges related to motivating teachers and students to delve deeper into OOP. Having a proper combination of interesting examples, projects and tools might help mitigate this.

Conducted in three separate workshops with participants from diverse high schools and study years, the combination of engaging topics, object-oriented programming (OOP), and game development demonstrated significant motivational potential in sparking interest in programming and STEM fields. Both students and lecturers provided positive feedback on the appeal of the topics, utilized technologies, and acquired skills. As a result of described activities it can be concluded that using games to teach OOP to high school students is a credible approach, because it allows and often even promotes: (a) integration of contemporary pedagogical and teaching practices to programming, (b) adoption of problem/project-based learning for a more realistic approach to programming, (c) placing an emphasis on student-centered learning, (d) student engagement and team collaboration, etc.

While performing analyses we came across different technologies that may be used for game-based OOP teaching, not all of these technologies were particularly suitable for high-school level. One of the technologies that stood out as a high-quality solution that is suitable for high-school level and supported by major players in industry (e.g. Oracle) was Greenfoot IDE. This tool was shown to be effective in facilitating gradual and visual learning of object-oriented programming and UML through the fun ways of game development. These conclusions are also supported by the results of the analysis of testing the use of Greenfoot in Gymnazium Pardubice. For this reason, we recommend it as a basis for creating a syllabus for teaching object oriented programming.

Finally, the culmination of our efforts has resulted in a set of innovative teaching and learning concepts, as evidenced by the comprehensive analyses conducted throughout this project result. The learning design methodology, detailed and defined in this chapter as well, serves

as a blueprint for planning and designing the implementation of the new approach. Leveraging teaching scenarios methodology as a powerful tool, we have vividly depicted diverse innovative approaches to education, creating a collection of teaching scenarios which are already inserted into the Learning Design tool. These scenarios, exemplifying our commitment to pedagogical innovation, are readily accessible for preview at http://learning-design.eu/en/preview/bd6a3cf90d09a6b6495ecf82/details. Furthermore, in an effort to share our insights and empower others, we have provided a template for defining teaching scenarios for entire courses, extending this valuable resource to our esteemed project partners responsible for PR3 and PR4. Together, these accomplishments underscore our dedication to advancing the landscape of education and fostering transformative and innovative learning experiences.

13. References

Abbasi, S., Kazi, H., & Khowaja, K. (2018). A systematic review of learning object oriented programming through serious games and programming approaches. *4th IEEE International Conference on Engineering Technologies and Applied Sciences, ICETAS 2017*, *2018-January*, 1–6. https://doi.org/10.1109/ICETAS.2017.8277894

An Integrated Environment for Learning Design. (2018, April 24). Frontiers. Retrieved May 3, 2023, from https://www.frontiersin.org/articles/10.3389/fict.2018.00009/full

Berssanette, Joao Henrique, & de Francisco, A. C. (2021). Cognitive Load Theory in the Context of Teaching and Learning Computer Programming: A Systematic Literature Review. *IEEE Transactions on Education*, *65*(3), 440–449. https://doi.org/10.1109/TE.2021.3127215

Berssanette, João Henrique, & de Francisco, A. C. (2021). Active learning in the context of the teaching/learning of computer programming: A systematic review. *Journal of Information Technology Education: Research*, *20*, 201–220. https://doi.org/10.28945/4767

Bottino, R. M., Ott, M., & Tavella, M. (2010). Empowering the Design and the Sharing of Learning Plans by Means of Net Technologies: The IAMEL System. In M. D. Lytras, P. Ordonez De Pablos, A. Ziderman, A. Roulstone, H. Maurer, & J. B. Imber (Eds.), *Knowledge Management, Information Systems, E-Learning, and Sustainability Research* (pp. 336–342). Springer Berlin Heidelberg.

Conole, G., & Fill, K. (2005). A learning design toolkit to create pedagogically effective learning activities. Journal of Interactive Media in Education, 2005(1), Art. 9.DOI: https://doi.org/10.5334/2005-8

Conole, G., Cross, S., Brasher, A., Weller, M., Nixon, S., & Clark, P. (2008). *A learning design methodology to foster and support creativity in design*. *978*.

da Silva, J. P., & Silveira, I. F. (2020). A systematic review on open educational games for programming learning and teaching. *International Journal of Emerging Technologies in Learning*, *15*(9), 156–172. https://doi.org/10.3991/ijet.v15i09.12437

De Assis Mota, A., Mota, L. T. M., & Morelato, A. (2004). Teaching Power Engineering Basics Using Advanced Web Technologies and Problem-Based Learning Environment. *IEEE Transactions on Power Systems*, *19*(1), 96–103. https://doi.org/10.1109/TPWRS.2003.821004

Examples of Learning Activities | Teaching & Learning | UTAS. (n.d.). Teaching & Learning. Retrieved May 3, 2023, from https://www.teaching-learning.utas.edu.au/learning-activities-and-delivery-modes/planning-learning-activities/examples-of-learning-activities

Hajdin, G., Vukovac, D. P., & Oreski, D. (2022). Redefining the e-schools concept of teaching scenarios for interdisciplinary topics. In EDULEARN22 Proceedings (pp. 5534-5542). IATED.

Hajdin, G., Hainš, V. V., & Oreški, D. (2018). The impact of teaching scenarios on student perception of teaching. In EDULEARN18 Proceedings (pp. 3305-3313). IATED.

Hendrik, H., & Hamzah, A. (2020). Flipped Classroom In Programming Course: A Systematic Literature Review. *International Journal of Emerging Technologies in Learning*, *16*(2), 220–236. https://doi.org/10.3991/ijet.v16i02.15229

Hundhausen, C. D., Agrawal, A., & Agarwal, P. (2013). Talking about code: Integrating pedagogical code reviews into early computing courses. *ACM Transactions on Computing Education*, *13*(3). https://doi.org/10.1145/2499947.2499951

IO2. Learning design models. (n.d.). TEACH4EDU4. Retrieved May 3, 2023, from https://teach4edu4-project.eu/en/intellectual-outputs/learning-design-models

Jawad, H. M., & Tout, S. (2021). Gamifying computer science education for z generation. *Information (Switzerland)*, *12*(11), 1–18. https://doi.org/10.3390/info12110453

Jenki, G. L., & Ademoye, O. (2011). Can individual code reviews improve solo programming on an introductory course? *ITALICS Innovations in Teaching and Learning in Information and Computer Sciences*, *11*(1), 71–79. https://doi.org/10.11120/ital.2012.11010071

Jerbić-Zorc, G. et al. (2021). Priručnik za primjenu i izradu e-Škole scenarija poučavanja. Hrvatska akademska i istraživačka mreža CARNET. Available at: https://edutorij.e-skole.hr/share/page/document-details?nodeRef=workspace://SpacesStore/b2c5cb3a-025a-4e2b-bbcb-6148613adab1

Luxton-Reilly, A., Simon, Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., … Szabo, C. (2018). Introductory programming: A systematic literature review. In *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*. https://doi.org/10.1145/3293881.3295779

Medeiros, R. P., Ramalho, G. L., & Falcao, T. P. (2019). A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education. *IEEE Transactions on Education*, *62*(2), 77–90. https://doi.org/10.1109/TE.2018.2864133

Perugini, S. (2019). Emerging languages: An alternative approach to teaching programming languages. *Journal of Functional Programming*, 1–13. https://doi.org/10.1017/S095679681900011X

Qian, Y., & Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education*, *18*(1), 1–24. https://doi.org/10.1145/3077618

Silva, L., Mendes, A. J., & Gomes, A. (2020). Computer-supported collaborative learning in programming education: A systematic literature review. *IEEE Global Engineering*

Education Conference, EDUCON, *2020-April*, 1086–1095. https://doi.org/10.1109/EDUCON45650.2020.9125237

Sharpe, R., Beetham, H., & De Freitas, S. (2010). *Rethinking learning for a digital age: How learners are shaping their own experiences*. Routledge.

Yulianto, B., Prabowo, H., & Meyliana. (2017). Effective digital contents for computer programming learning: A systematic literature review. *Advanced Science Letters*, *23*(5), 4733–4737. https://doi.org/10.1166/asl.2017.8877