Microlearning and Automated Assessment – A Framework Implementation of Dissimilar Elements to Achieve Better Educational Outcomes

Ján Skalka

1 Introduction

Writing the source code of programmes is currently one of the basic skills of a modern employee. Many support systems of various levels, content and quality have been created to support the teaching of programming (Crow et al., 2018). Many researchers seek to focus on the narrow field of programming as such and explore the modern learning environment in a broader context, often in the interconnection of STEM/STEAM area (Çetin & Demircan, 2020; Smyrnova-Trybulska et al., 2017).

The research trends of the last few years are aimed at predicting success or failure in education (Kabathova & Drlik, 2021; Drlik & Munk, 2019). However, the most crucial element of education is the content, form, and distribution to the student (Carlon et al., 2020). Several frameworks have been designed and implemented in recent years to optimise content, distribution, and retain student attention (Halvoník & Kapusta, 2020; Sharma et al., 2012).

Mobile applications are gradually becoming the most important distribution channel due to their ease of use and availability anytime and anywhere (Baldwin & Ching, 2020). The use of mobile applications in education, research in the field of personalisation (Moon et al., 2020; Morze et al., 2021; Bartolomé et al., 2018) and monitoring of user behaviour (Halvoník & Kapusta, 2019) has also intensified.

The article deals with the search for an answer to whether it is possible to combine two effective approaches in teaching programming – microlearning and automated assessment. Methodologies of their use are developed in many sources, and their isolated use is currently a frequent subject of pedagogical research. However, the combination of both approaches is unique and represents an additional

J. Skalka (✉)
Constantine the Philosopher University in Nitra, Nitra, Slovakia
e-mail: jskalka@ukf.sk

E. Smyrnova-Trybulska et al. (eds.), *Microlearning*,
https://doi.org/10.1007/978-3-031-13359-6_1

combination of obtaining basic information and its practical use for writing programs in different programming languages.

In addition to the design and presentation of an educational environment combining learning in small units and tools designed for automatic evaluation of source codes, the article also includes evaluating the perception of the environment and the educational approaches used by students.

The article main aim of the article is to present architecture, current state, and experience with the pilot deployment of virtual learning environment *Priscilla* (Skalka & Drlik, 2018a, b), developed based on the conceptual framework for teaching and learning programming (Skalka et al., 2021).

This environment effectively combines contemporary promising educational approaches, including microlearning (Hug, 2005) and automatically evaluated source codes (automated assessment) (Ala-Mutka, 2005; Fernández Alemán, 2011). The balanced combination of these approaches allows effectively managing the time required for learning theory, applying the obtain knowledge immediately, minimising the time for source code evaluation, and providing immediate feedback, which is important for learning programming.

The research questions are defined as follow:

- RQ1: *What is the effective software architecture covering the needs of the framework defined for learning and teaching programming in introductory courses.*
- RQ2: *How do students perceive the methods of microlearning, and how, according to them, does it contribute to the improvement of their programming skills and knowledge.*
- RQ3: *How do students perceive the method of automated assessment, and how, according to them, does it contribute to the improvement of their programming skills and knowledge.*

The article has the following structure. The second part summarises information about selected information systems for teaching programming and web portals used in programming learning. The third part presents the introductory conceptual model and implementation of the backend and front-end parts. This section also describes the most important framework modules implemented in the *Priscilla* system. The fourth chapter deals with studying the perception of the system by students who completed one semester of study. Finally, the article concludes with a discussion, a description of the current state and future work.

2 Introductory Programming Learning Environments

Despite the relatively extensive research in introductory programming courses, the specific research focused on developing proprietary solutions used by universities is rare. Many universities use plugins or modules implemented in Learning Management Systems (LMS). Skalka et al. (2019) used the LMS plugin

implemented by Rodríguez-del-Pino et al. (2012) for LMS Moodle to support automated evaluation of source codes in the introductory programming course of Java.

The following examples of original solutions and software systems for the teaching of programming are considered very promising.

Vesin et al. (2018), Blaženska-Tabakovska et al. (2017) presented Programming Tutoring System (ProTuS) with a cross-platform architecture that aggregates and harmonises study analyses from different systems and quantifies student performance through a set of indicators. Learning is based on a combination of explanations, interactive examples, interactive challenges and coding exercises.

Brusilovsky et al. described the use of the Python Grids System (Brusilovsky et al., 2018) as a tool that provides access to four types of interactive tutorial content for learning Python: annotated examples, animated examples, semantic code evaluation problems, and code construction problems.

Buffardi & Edwards (2014) introduced CodeWorkout – an online training system with course management functions. It hosts an online repository of questions and assignments that teachers can incorporate into their courses. It also provides tools for creating new items so that the exercises can be adapted to the class's needs.

Many courses provided through MOOC portals such as Coursera, Edx, Udemy often contain various types of "camps" that allow writing, running and evaluating codes, either at the automatic level or through peer-review (Chauhan, 2014; Johnston, 2015).

University solutions are complemented by various categories of public portals and applications which offer free courses for the public and life-long learning. Each of them is specific, often closely oriented on technically skilled students without implemented standard didactical methodology. The simplest category of portals provides an essential source of information, where the popular *w3schools.com* was chosen as a typical example. The second category covers portals supporting the development of programming skills by writing programs with the support of many programming languages. Here it is assumed that the user already has basic knowledge and educational content is usually not available (*Hackerrank*, *Codewars*). The next category consists of portals providing content in the microlearning form with various types of competitions. It is assumed that the user achieves the course goals based on internal motivation, ensured by various competitions and strong gamification (*Sololearn*). The last category is represented by portals intended for the youngest users. They can replace writing code by automatically entering entire commands or block-based language depending on age.

Table 1 compares the presented *Priscilla* portal, as a portal based on microlearning and automated assessment with other solutions.

In addition to the portals listed in Table 1, which offer educational content for multiple programming languages, many other portals are focused on a specific programming language. Many solutions make it possible to integrate selected parts of the content into teaching or use web portals as a suitable supplement for practising educational content.

Table 1 Popular free web portals focused on programming learning compared with a real implementation of the presented framework by system PRISCILLA

Portal/property	w3schools.com	codewars.com/ qualified.io	sololearn.com	freecodecamp.org	hackerrank.com	codeavengers.com	code.org	PRISCILLA
Age category	Teens, adults	Teens, adults	Teens, adults	Teens, adults	Teens, adults	5+	4+	Teens, adults
Supported languages	JavaScript, HTML/CSS; PHP in simple form	All language types	All language types	JavaScript, HTML/CSS, python	All language types	JavaScript, HTML/ CSSweb languages, python	Primary block-based visual programming	All language types
Content	Basic	–	Yes	Basic	As part of tasks	Yes	In a specific form	Complex in microcontent
Micro-content	–	–	Yes	–	–	–	–	Yes
Quizzes	Basic	–	Yes	–	–	Yes	–	Yes
Automatic code evaluation	Yes	Yes	–	Yes	Yes	Yes	Yes	Yes
Sandbox or own code space	Yes	–	Yes	–	–	–	–	In preparation
Learning paths/ courses	Basic	–	Yes	Yes	Yes	Yes	Yes	Yes
Competitions	–	Yes	Yes	–	Yes	–	–	Yes
Gamification	–	Yes	Yes	–	Yes	Yes	In a specific form	Yes
Teaching	–	Yes	Create content	–	–	Yes	In a specific form	Yes

3 Learning Environment Concept

Successful and sustainable implementation of the framework requires coverage of introductory programming courses and activities intended for future educational environment development and content development. Taking care of content updates and creation and updating design following modern design trends can be covered by educational activities in advanced engineering courses. Students will work on the development of an environment that they know because they studied in it the basics of programming.

The implementation of the framework (Skalka & Drlík, 2018a, b) defines the concept and learning processes into independent systems preceded by the implementation in the *LMS Moodle* environment (Skalka et al., 2021). Typical tests in *Moodle* with quiz questions of various types were used to cover the needs of microlearning. Prepared tests consist of simple answers through the selection of options to complete the source code. Automatic code evaluation was provided by the *Virtual Programming Lab* supporting automatic source code evaluation in many programming languages (Rodríguez-del-Pino et al., 2012).

Using *Moodle* during implementation has resulted in the need to address many limitations and did not produce the expected results in the user interface. The most problematic places were the static structure of the course, which does not support the efficient display of a large number of course objects and the complicated integration of gamification elements into the system. Support for user activity logging and support for learning analytics, which are the essential features of a system for understanding the learning process, did not provide detailed information on user behaviour. It has also been laborious for users to obtain detailed information about fixes and source improvements. The ability to adapt the user's view of the educational content was low, etc.

The form of programming new modules in *LMS Moodle* is precisely given, and module programmers require a thorough knowledge of the LMS system and the use of spaghetti code in *PHP*. The complicated development has significantly reduced the potential for sustainable system development due to lower motivation and higher demands on students in advanced programming courses.

The logical step was to create a stand-alone, fully adaptable system in-house that primarily supports the requirements of the framework and is based on new popular and widely used technologies.

Following the positive experience with microlearning activities and exercises based on automated source code evaluation in *LMS Moodle* (Skalka & Drlik, 2020) and requirement of conceptual design presented above and in (Skalka & Drlik, 2018a, b), the concept of a software architecture proposal of a system called *Priscilla* (PRogressIve System for interaCtIve (programming) Learning and Learning Assistance) was designed. Its structure and implementation are presented in this section.

3.1 Framework Architecture

The conceptual model of *Priscilla* presented in Fig. 1 is structured as three-layered architecture, which contains an independent front-end part (presentation/client module) and separate backend parts integrated into the server infrastructure. The communication between front-end and backend is realised via the API interface, and particular features use web sockets.

The front-end part can be implemented as a web, mobile or desktop application. The user's interaction with the application is fluent because the network traffic is very low after the first application launch in a web browser.

The front-end part provides the educational content in three forms:

- Micro-content represents the content in the form of text, short source codes, images, etc. This type of activity is designed as an HTML container, and the content is transmitted as a package containing formatted text (headings, text, source code, images, tables, etc.).
- Microlearning activities are interactive objects that require the user to solve simple tasks. A typical example is filling in the correct code result, filling a gap in the code by typing or drag-and-dropping the right parts, reordering shuffled lines of source code, and so on. Interactive activities are combined with content activities (usually 1:1 or in favour of interactive activities) in lessons and chapters. Tasks in interactive activities are focused on the information contained in previous content activities – the content structure is developed concerning microlearning principles.

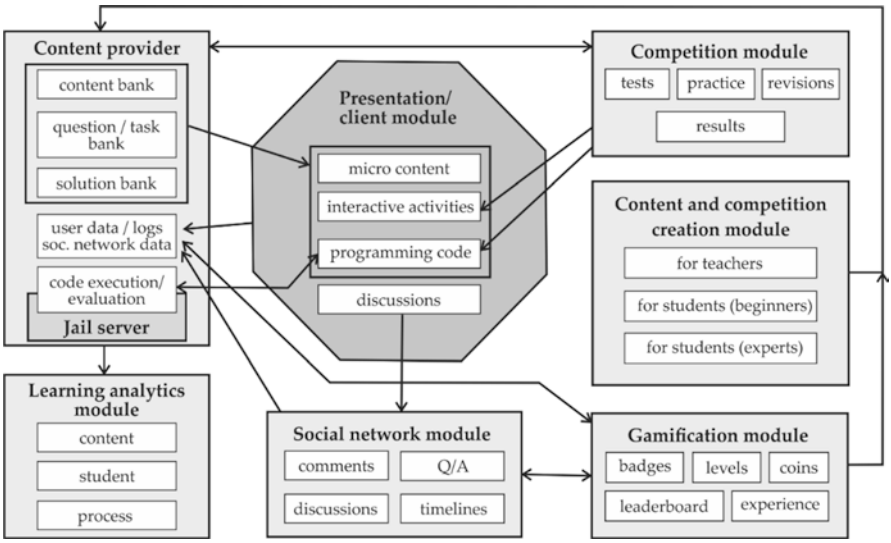


Fig. 1 Simplified conceptual model of PRISCILLA-2.0 based on the PRISCILLA model presented in (Skalka & Drlik, 2018a, b)

- Activities aimed at acquiring programming skills are focused on writing, executing and validating the program code. The student completes the developed programs or writes complete codes in a user-friendly editor adapted to the selected language. After writing the code, the student sends the program to the validation system, which evaluates its correctness. The response may contain compiler errors (syntax errors) or code accuracy, which depends on comparing the submitted code results with the expected results.

The front-end part allows the student to use the discussion module to communicate with their classmates, rate the content and activities and report errors or inaccuracies in the content. Each user's action causes a connection to the API interface and records the action type and user identification. Many activities require an educational system response implemented by RPC (Request-Response Protocol) using the JSON format.

Responses are generated on the backend part, which is divided into two physical and several logical segments. Two independent systems present the physical parts:

- The educational system is implemented as a web application working with data stored in a database system. This structure will be described later.
- The jail-system is implemented as an independent Linux system designed to verify the source code. Because program code verification is often based on program execution, the system must be resistant to attacks, malicious code, and system errors and must be self-healing. The *Priscilla* system uses the jail-server developed for the *Virtual programming Lab* in Moodle (Rodríguez-del-Pino et al., 2012), which can evaluate dozens of programming languages. The jail-system creates a new temporary user with low privileges for every task, and after reading the results, the user is removed from the system. The restrictions defined for program activities are derived from *Linux* user permissions.

The logical structure of the backend reflects the education system functions and the ideas presented in the previous section. It is designed so that the individual parts cover all the functions of the system. The parts are closely linked with each other, as activity in one part often causes related activity in the other part. The backend has the following components:

- *The Content provider* provides access to all educational content. The main part of the content is divided into lessons and chapters organised in educational courses. Extended content is intended for tests, exercises, revisions and competitions. Each question, task or assignment is accompanied by tips, hints and correct answers or authoring solutions of the programs. *The Content provider* processes the requests from the client interface and sends the content or evaluation results. All evaluation algorithms are implemented in the backend part to prevent hacker attacks. The *User data module* is a part of the *Content provider* containing information about all activities, attempts, and users' results in the system. This part of the data is primarily intended for the *Learning analytics module*.

- *The Content and competition creation module* is determined for content building. This section is intended for administrators or content creators, and the typical user is not authorised to use the features of this module. The module provides functions for competitions, courses, chapters and lesson structure creating. Content, questions and assignments can fill built elements.
- *The Competition module* ensures the realisation of activities aimed at testing students (in organised education) or competitions of students with each other. It offers prepared content in educational objects (matches, tests, revisions, etc.) and keeps track of time defined for them. The module also includes the evaluation of test results as a whole and the ordering of competitors. The structure of the questions is identical to the items used in the learning part. Two main areas are used in competitions – users can compete in answering questions or writing programs (rated for writing speed, execution speed, or code effectiveness).
- *The Social network module* is a layer that provides task-related discussions, commenting, micro-object evaluation, bug reporting, and general discussion management. Each discussion post can be evaluated (positively or negatively), and the author can get feedback, which is also used in the gamification part.
- *The Gamification module* monitors user activities and processes the collected data into gamification elements. The most frequent gamification components of the *Priscilla* system are badges in many categories (different types of experience with the learning process, experience with competition, evaluation, and activities in discussions, contribution to the system, etc.). Badges are also graded according to performance into several levels (bronze, silver, gold, diamond, etc.). Each action in the system triggers event processing in the *Gamification module* and changes the monitored user parameters.
- *The Learning analytics module* is designed to analyse and evaluate the user's behaviour and educational outcomes, identify problematic parts of the content and predict the user's preferences and success. This module does not create new data; it only processes the data of *the Content provider* and displays it based on the teacher or administrator's defined views. The module helps to tune and optimise the parameters of the system.

3.2 Backend Implementation

Typical attributes of modern software systems are permanent availability, fast processing of many parallel requests, and orientation to the data provided through services. Complex systems usually consist of related services that work independently and can be developed in isolation. Increased flexibility gained by adopting paradigms such as API-oriented architecture is associated with creating robust and complex systems (Brosig et al., 2014). The communication between the front-end and the backend is provided via web services. This architecture allows the development of various front-end applications: web-client, mobile application, or desktop application.

The core of the *Priscilla* system based on the conceptual model is implemented as a server application developed in the PHP framework *Laravel Lumen* intended to develop applications based on microservices. The current database system is *MySQL*. The communication is realised via REST API using application/JSON format.

The backend part of the system processes front-end requests in several layers and is depicted in Fig. 2:

- The first layer verifies the user's identity. Only the requests of the authorised and logged-in user will be moved for further processing. Authentication is provided by OAuth components (Ferretti et al., 2017).
- The API layer identifies the request and selects the correct service to process or provide the data.
- Service is usually a single-purpose method for providing communication with a database or simple request processing. The services can be combined and typically write a record of the operation in a database recording the user's behaviour and results.

Services can be divided into three types: services for processing anonymous activities (login, registration, visits to the main page of the system, etc.), activities with

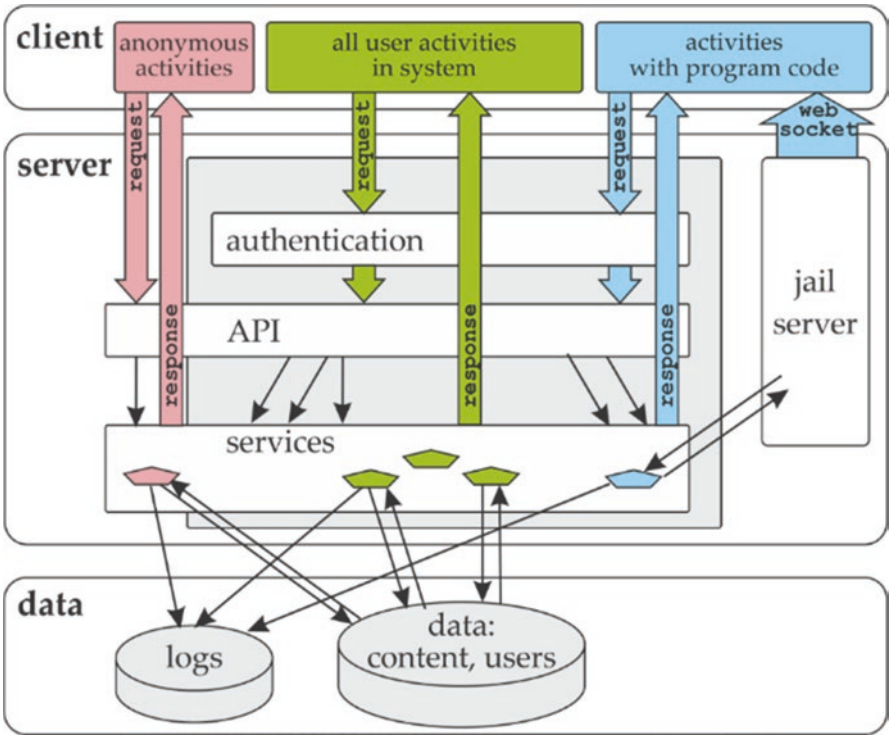


Fig. 2 The software structure of PRISCILLA implementation based on microservices

program codes that are specific and all other activities performed by the logged-in user in the system.

Anonymous activities skip the authorisation layer and process requests directly. The answer may also include data from the database.

Activities with automatic source code evaluation are specific because it is necessary to ensure communication with the jail server. The communication of the application as a whole with the jail-server is realised as follows:

- the user in the front-end asks to check the correctness of his program,
- the service invoked in the backend stores code of the delivered program into the database and prepares the request to the jail server,
- the backend sends a request to the jail-server and, in response, immediately receives a token representing the jail-server process executing the source code,
- the obtained token is sent as a response (to the demand of the code verification) to the front-end,
- the front-end gets a token and opens a web-socket to the jail-server; jail-server has meanwhile started the execution of the program delivered from backend,
- front-end reads the changes on the jail-server via the socket, and if the jail-server reaches one of the final states (error, long program execution time, program completion, etc.), the front-end sends a request to the backend to read the results,
- the jail-server results are read by the backend service and written to the database; at the end of the process, the service sends evaluation results to the front-end.

The process is a bit complicated due to the decrease of server load and the elimination of cheating. Direct communication with the jail-server is realised only on the backend. The time-consuming operation of monitoring the running program's activity on the jail-server is again implemented on the client side.

All other activities are carried out uniformly: After defining the application client's request parameters and calling the appropriate microservice, the backend realises user authentication, authorisation verification (user, admin), and subsequent request processing. The standard services cover common CRUD operations, evaluation of the solution's correctness, logging of activities, gamification and use of social network elements. Task evaluation is performed exclusively on the server to eliminate cheating.

3.3 The Front-end Implementation

The current version of the application's front-end part was developed in the *VueJS* environment with the definition of the appearance based on the rules of *Material design* of *Google*. The system is designed to teach many programming languages, and the structure of the system supports their teaching in one application. The example of used courses, languages and user interface is presented in Fig. 3.

Language support depends primarily on language interpreters (compilers) and then on advanced content (defined usually by content developers or teachers). Each

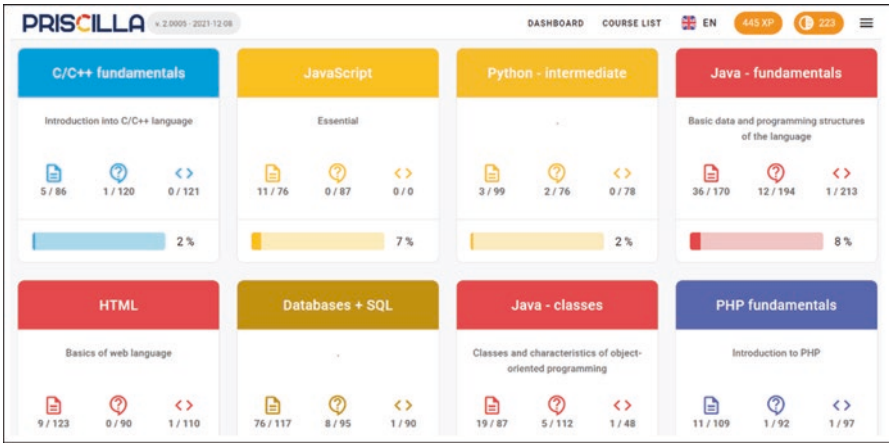


Fig. 3 The user's view of the dashboard and his opened courses in the PRISCILLA system

language has defined a default lesson path consisting of microlearning activities (tasks) and programming tasks (code).

3.3.1 Content Structure

The essential idea in successful introductory programming courses is to leave students some freedom to choose the order of activities they should complete in the programming course. The programming courses were designed following the classical educational structures, and the order of defined chapters is in line with the didactics of teaching programming. Still, they do not force the student to proceed linearly. Almost every chapter contains a combination of tasks and programs, which students complete based on their preferences. Each task can be repeated as many times as a student needs. Students can return to the place of explanation of the issue, if necessary – the system's goal is not to evaluate but to teach.

The basic information displayed to the student is the progress of completed questions and submitted programs that are part of each chapter. An overview of the open course Java – fundamental (Skalka et al., 2020a) is shown in Fig. 4.

Each chapter displays an icon indicating whether it should be started or whether the user should solve more tasks in the previous chapter. The recommendation is calculated to a 50% success rate of tasks and programs in the previous chapter. No chapter is locked; there are only recommendations, and the user can study any chapter at any time.

The panel on the right side contains information about the last completed activity in the course, the achieved score and the amount of currency gained, and other gamification objects.

All interactive activities are dynamically generated based on data obtained from the backend part of the system and a standard universal template.

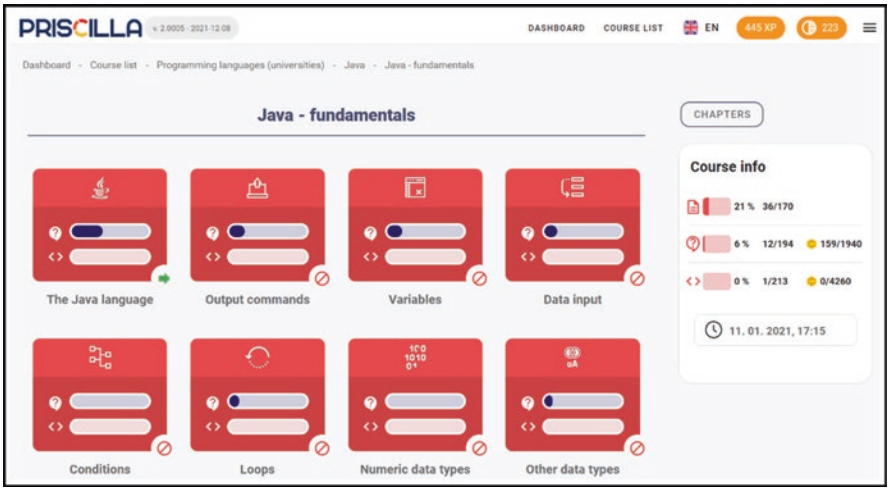


Fig. 4 User's view of the Java course content

A combination of micro-content and micro-tasks realises the implementation of microlearning in the system. The micro-content contains brief information, and the micro-task follows it and contains a question ensuring the repetition or consolidation of the presented information. It is advisable to alternate micro-content and micro-tasks within the lessons in a ratio of 1:1 or more (one content and at least one task). The specific content of micro-content and micro-task are presented in Figs. 5 and 6.

Support for building skills in several ways is based on a combination of different types of tasks. There are available the following task types covering the following activities:

- typical domain verification tasks (short answer, choice of options),
- placing code snippets,
- supplementing the writing of commands or parts of code,
- finding the results of subroutines,
- rearranging lines of source code,
- different types of writing programs (in whole or part).

3.3.2 Automatic Source Code Evaluation

Exercises based on automatic source code evaluation consist of three basic types.

The most used and simplest type for the content creator automatically evaluates programs based on comparing the program's correct outputs with the outputs of the user program (I/O approach). The definition of evaluated inputs has a typical structure compatible with the definition of inputs and expected results in the VPL

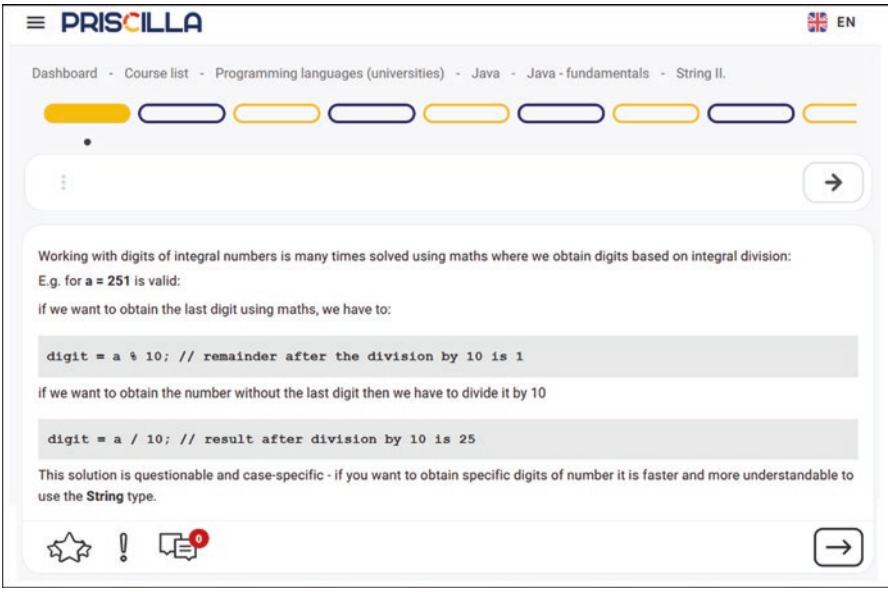


Fig. 5 Example of micro-content in the educational system PRISCILLA

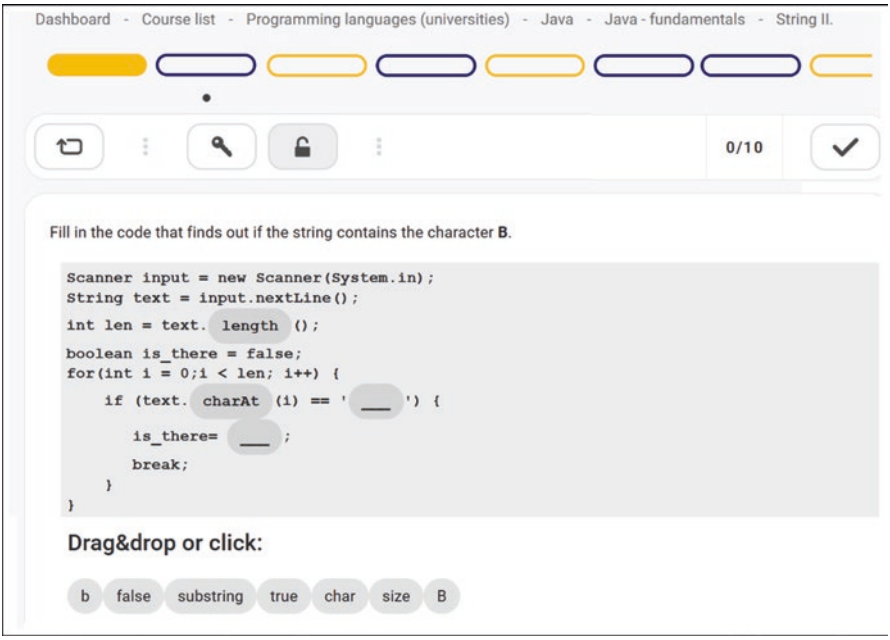


Fig. 6 Example of micro-task in the educational system PRISCILLA

environment (Rodríguez-del-Pino et al., 2012). The example of test cases and their use is presented in Figs. 7 and 8.

Based on *xUnit* testing ideas, the second approach is typical for tasks designed to learn object-oriented programming. It uses automated tests principles, and its implementation depends on the creators' abilities and habits. Each content creator can implement their testing methods. The easiest way is to use the *xUnit* libraries, where the creator has set the tested methods and the correct outputs.

The system is also open to unique approaches. The content creator can create his random generators for selecting a sequence of methods, selecting input values, and using the author's solution as a sample solution, with the results of which the student's solution will be compared.

An example of a particular class used for program evaluation in the form of another class defined in the assignment is shown in Fig. 9. The definition uses an input matrix that will be set as attributes of class instances passed by students. Each attribute and method should be tested for random and threshold values. The user output has the same design as the *Execution info* section in Fig. 8.

The last type of automatic evaluation is a static evaluation used in any programming language of varying complexity and difficulty. Its simple version based on content (not structure) analysis is used, for example, in HTML courses. The idea is based on defining the rules and evaluating their fulfilment. *Priscilla* contains several rules that can be used to varying degrees to validate a document (text). The rules are defined to check the existence, position, or order of text patterns. A simple example is presented in Fig. 10.

```
case = Test1
input=here-there
output=10
case = Test2
input=MOTHER.
output=7
case = Test3
input=Winter is coming.
output=18
case = Test4
input=springspringspringspringspringspring
output=36
```

Fig. 7 Test cases definition for code that should print the number of characters in a defined string

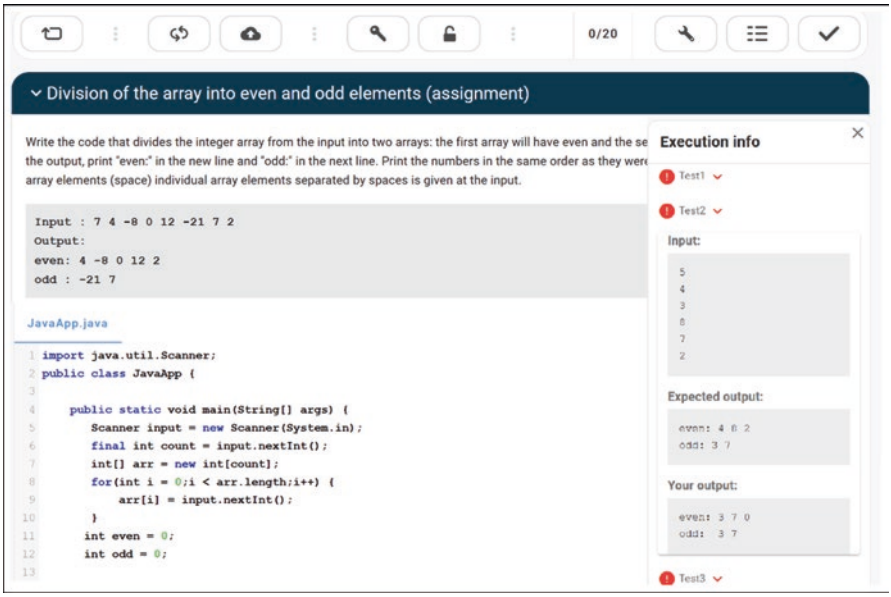


Fig. 8 The result of program code evaluation in the implementation of the presented framework in the educational system PRISCILLA. The Execution info section shows the inputs and outputs of test cases in which the expected and obtained values do not match

3.3.3 Learning and Teaching Support

For each task, the template provides the ability to invoke help or unlock an answer, add a discussion post, report errors, and rate the quality of the question. The activities dedicated to programming are extended by sending the program to evaluate and display compiler messages or test results. The views of activities are presented in Fig. 11.

The user interface for competitions (test, revisions etc.) uses the same templates and activity types, but the time to prepare answers for tasks and programs is limited. After the set time has elapsed, all the tasks (including unfinished ones) are automatically submitted and evaluated.

The educational environment includes gamification tools – levelling, awarding badges, rewarding selected activities and rankings for individual courses or programming languages.

A teacher role has been created in the *Priscilla* system to support the use of blended activities. This role can be acquired by any user who sets up a study group, where the students join by the key. The teacher has permission to monitor students activity and results in his group, and he can participate in solving course activities.

```
public class Evaluate {

    public String doTests(Object[][] cases) {
        String output = "";
        int correct_count = 0;
        for(int i = 0; i < cases.length; i++) { // number of tests, test cases
            int value1 = 0;
            int value2 = 0;
            switch (((String)cases[i][0])) {
                case "P": value1 = (int)(Math.random()*1000); break;
                case "N": value1 = -(int)(Math.random()*1000); break;
                case "Z": value1 = 0; break;
                case "R": value1 = -100000+(int)(Math.random()*200000); break;
            }
            switch (((String)cases[i][1])) {
                case "P": value2 = (int)(Math.random()*1000); break;
                case "N": value2 = -(int)(Math.random()*1000); break;
                case "Z": value2 = 0; break;
                case "R": value2 = -100000+(int)(Math.random()*200000); break;
            }
            // tested class and author solution
            MyClass tested = new MyClass();
            MySolution correct = new MySolution();

            int result_test = tested.getMax(value1, value2);
            int result_correct = correct.getMax(value1, value2);
            // results comparison
            if (result_test != result_correct) {
                output = output + "!--- result: " + TEXT_ERROR + "\n";
            } else {
                output = output + "!--- result: OK\n";
                correct_count++;
            }
           ...
```

Fig. 9 A simple example of a class designed to compare the results of students classes with the original solution. The assignment was simple – create a method for the sum of two real values. Test cases are defined by string constants – P (positive values), N (negative values), R (random values) and Z (zero). The randomisation of input values minimises the risk of false positives

4 Students' Perception of the Elements of the Priscilla System

Priscilla was first deployed in the winter semester of 2020/2021 as the primary teaching tool for Java courses and a complementary database and SQL learning tool. Other courses were used to support additional activities in the voluntary preparation of students.

The research focused on the perception of elements of the system by students was carried out after the end of the semester. Answers of the Java course students

Fig. 10 Example of static automated evaluation of HTML program code in the implementation of the educational system PRISCILLA. The rules are defined in the admin interface, and the user views only a simple window after evaluation

were collected not anonymously to find dependency between students results and their perception of the educational process. The questionnaire was focused comprehensively, the coverage of the topics of lessons by micro-content and automatically evaluated programs was identified. A series of similar questions focused on perception by students was devoted to individual elements of the environment.

The questionnaire respondents were students of the first year of the study program of applied informatics aged 20–23 years.

Table 2 presents the perceptions of micro-lessons by students. The course content was created to evenly cover all the topics covered in the introductory course of programming. The perception of the compliance of the content of micro-lessons and lectures realised in 2020 in online form expresses mastery and understanding of content by students. If students perceive that the taught and the practised content are the same, they are likely to understand the context or at least paid sufficient attention to the content. The first question in the questionnaire finds out this fact.

The second group of questions focuses on identifying the role of micro-content through the Likert scale. The role of micro-content is expressed in questions at different levels:

- Micro-content and micro-questions helped students understand the curriculum.
- Micro-content and micro-questions helped students practice previously understood curriculum content.

(a)

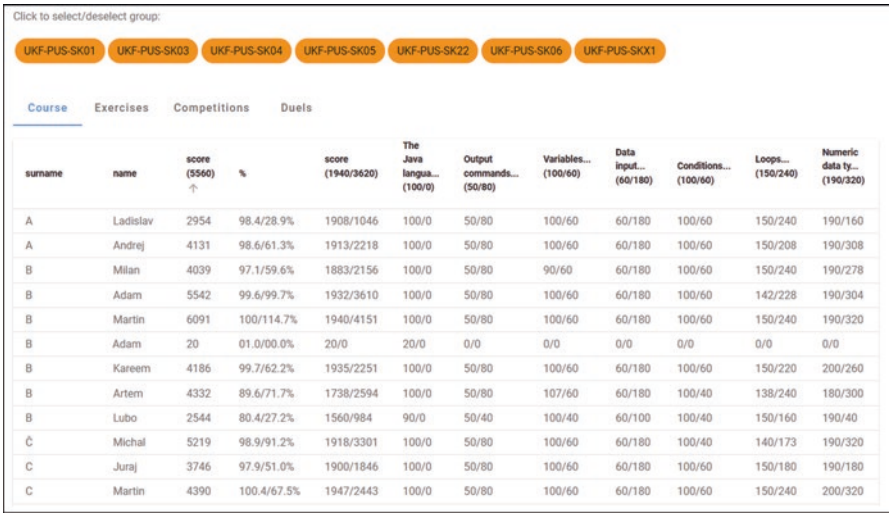
(b)

Fig. 11 Functions implemented in interactive (**a**) and programming (**b**) activities. There are shared functions on the header toolbar – get help, buy the correct answer and in the program activity: show/hide the compiler message and show/hide the execution information. The footer of each activity contains icons for rating assignments, bug reports, and discussion views

- Micro-content helped students with a comprehensive mastery of the curriculum – the student used it as a primary source of learning.

It can be observed that majority of students perceived microlearning positively to very positively.

Table 3 presents the perceptions of automated assessment by students. The most important characteristics of educational content are students understanding and the teacher's (or course creator's) ability to assign a task clearly and accurately. The first question focuses on identifying the unambiguity and comprehensibility of the assignment.



surname	name	score (5560)	%	score (1940/3620)	The Java langua... (100/0)	Output commands... (50/80)	Variables... (100/60)	Data input... (60/180)	Conditions... (100/60)	Loops... (150/240)	Numeric data ty... (190/320)
A	Ladislav	2954	98.4/28.9%	1908/1046	100/0	50/80	100/60	60/180	100/60	150/240	190/160
A	Andrej	4131	98.6/61.3%	1913/2218	100/0	50/80	100/60	60/180	100/60	150/208	190/308
B	Milan	4039	97.1/59.6%	1883/2156	100/0	50/80	90/60	60/180	100/60	150/240	190/278
B	Adam	5542	99.6/99.7%	1932/3610	100/0	50/80	100/60	60/180	100/60	142/228	190/304
B	Martin	6091	100/114.7%	1940/4151	100/0	50/80	100/60	60/180	100/60	150/240	190/320
B	Adam	20	01.0/00.0%	20/0	20/0	0/0	0/0	0/0	0/0	0/0	0/0
B	Kareem	4186	99.7/62.2%	1935/2251	100/0	50/80	100/60	60/180	100/60	150/220	200/260
B	Artem	4332	89.6/71.7%	1738/2594	100/0	50/80	107/60	60/180	100/40	138/240	180/300
B	Lubo	2544	80.4/27.2%	1560/984	90/0	50/40	100/40	60/100	100/40	150/160	190/40
Č	Michal	5219	98.9/91.2%	1918/3301	100/0	50/80	100/60	60/180	100/40	140/173	190/320
C	Juraj	3746	97.9/51.0%	1900/1846	100/0	50/80	100/60	60/180	100/60	150/180	190/180
C	Martin	4390	100.4/67.5%	1947/2443	100/0	50/80	100/60	60/180	100/60	150/240	200/320

Fig. 12 Monitoring of student activities in teacher defined groups

Table 3 Perceptions of automated assessment by students in the winter semester of 2020/2021

Question	1 (disagree)	2	3	4	5	6	7 (agree)
Clarity and accuracy of assignments	4	2	8	15	**18**	**18**	**10**
Help to understand	3	1	6	9	**16**	**17**	**23**
Practice understood content	4	0	2	10	**13**	**22**	**24**

Table 2 Perceptions of micro-lessons by students in the winter semester of 2020/2021

Question	1 (disagree)	2	3	4	5	6	7 (agree)
Compliance of micro-lessons and lectures	4	0	2	9	**8**	**27**	**25**
Help to understand	3	0	7	11	**13**	**21**	**20**
Practice understood content	3	2	4	8	**11**	**23**	**24**
Primary source of learning	2	4	5	17	**16**	**17**	**14**

The following questions focus on identifying the role of automated assessment through the Likert scale again. The roles of automated assessment were expressed at two levels:

- The automated assessment helped students understand the curriculum.
- The automated assessment helped students practice previously understood content.

The majority of respondents perceived automated assessment positively.

The results of two continuous tests aimed at identifying students' ability to write entire programs independently were used to inspect the relationship between students' answers and their learning outcomes. The maximum score of this pair of tests was 1000 points (500 per test). The histogram in Fig. 13 presents the distribution of

the results. Questionnaire respondents who were evaluated in a different way (external study) were omitted from the sample.

The correlations between students' results and the answers to the questionnaire questions are presented in Table 4. The dependence was identified using the Pearson correlation coefficient.

The dependence between characteristics is proven in the case of a value greater than 0.4. The evaluation results demonstrate that there is no dependence between the results of students and their perception of individual types of educational objects.

5 Discussion

The answers to the research questions can be summarised as follows

- RQ1: *What is the effective software architecture covering the needs of the framework defined for learning and teaching programming in introductory courses.*

The software architecture was designed to be able to cover the needs of the framework defined in (Skalka & Drlik, 2018a, b) and at the same time bring a user and research design that is better than its implementation presented in (Skalka et al., 2021). The essential feature of the system is open to any front-end implementations covering the creation of the web, mobile and desktop applications on the same back-end kernel.

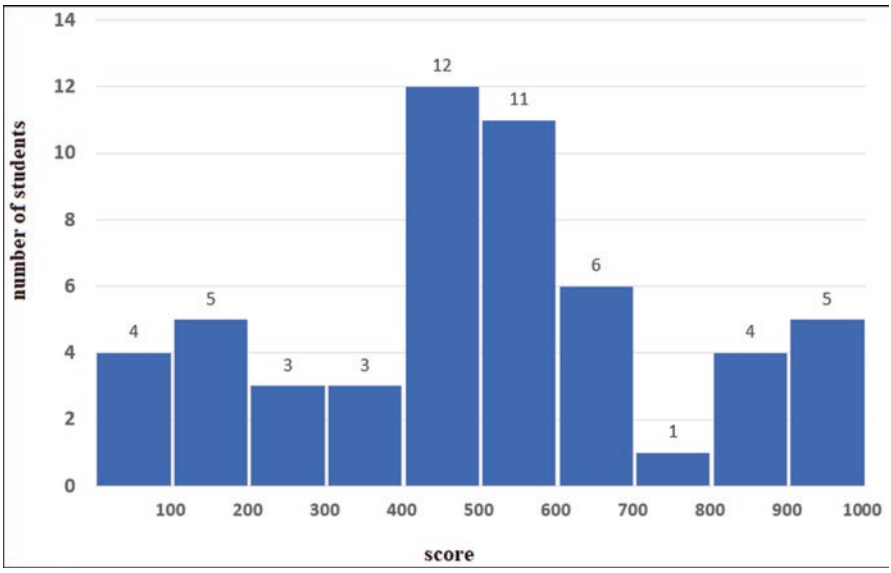


Fig. 13 Histogram of student test results

Table 4 Correlations between student results and questionnaire questions

Activity/questions	Pearson correlation coefficient
Compliance of micro-lessons and lectures	0.22
Micro-lessons helped to understand	0.17
Micro-lessons helped to practice the understood content	0.12
Micro-lessons were a primary source of learning	0.01
Clarity and accuracy of assignments in automated assessment	0.26
The automated assessment helped to understand	0.35
The automated assessment helped practice understood content	0.26

The functionality of the backend kernel ensures the control of responses and the evaluation of source codes. This approach is standard, implemented in several similar systems (Chen et al., 2018; Silva et al., 2017) and, thanks to its isolation from user activities on the front end, also relatively secure (Liebenberg & Jarke, 2020).

Based on conceptual framework ideas, the presented system covers an educational concept implemented as an essential tool for teaching programming at five European universities. The educational system is used to cover the first framework phases defined for building knowledge and skills. *Priscilla* provides an environment to offer content availability, instant feedback in all types of assignments, the ability to communicate between users, and the support of a full-time study of learning programming.

It covers many activities needed to educate programmers at novice levels. There are 24 courses in 8 languages implemented to cover Java, Python, C, HTML, CSS, JavaScript, PHP and SQL. Every course is localised into English, Spanish, Slovak, Czech and Polish languages.

Currently, the system has about 1500 unique active users, so it can be concluded that the proposed concept is functional and successful.

- RQ2: *How do students perceive the methods of microlearning, and how, according to them, does it contribute to the improvement of their programming skills and knowledge.*
- RQ3: *How do students perceive the method of automated assessment, and how, according to them, does it contribute to the improvement of their programming skills and knowledge.*

A pair of research questions were answered through a questionnaire with the following results based on the evaluation of the feedback obtained after the end of the semester on a sample of 75 first-year students. It can be stated that:

- micro-lessons **help students understand** new content and are sufficient for 72% of students as a basic source of information, 13% disagree with this statement,
- 77% of respondents say that microlearning **helped them practice** educational content, and 12% are negative about the claim,
- 63% of students can accept micro-lessons as the **primary source** of information when learning programming, 15% disagree with the statement,

- automated assessments help students **understand** new content for 75% of students, 13% disagree with this statement,
- automated assessments help students **practice** content and are sufficient for 79% of students, 8% disagree with this statement,

The dependence between students' educational results and their perception of micro-content and automated assessment has not been proven. This finding is quite important because it does not favour micro-lessons or automated assessment only for a selected group of students. Statistically, students with better and students with weaker results perceive it in the same way.

6 Conclusion and Future Work

The next phases of students' education focus on developing advanced skills and knowledge use the education system as an environment whose content and modules can students develop. As part of the verification of the framework concept, the following activities will be implemented in the next part of their study:

- Students will be involved in creating new questions and tasks after completing the introductory courses. Creating new assignments expands the educational content provided by the system. Discussion and analysis of new content will create an area for students better to understand the relationships between elements of their acquired knowledge. Self-expression skills and skills for building code and writing test classes or scripts will also be improved. This activity will be covered by students' obligation to create new tasks and provide tools for their verification within the advanced subjects dedicated to application development. Feedback and quality evaluation of the new elements will be provided on two levels. The first level will be covered by user evaluation, which is a part of all micro-units. It will be a subjective part of the evaluation. The evaluation's objective aspect will be realised by learning analytics tools, which can identify outliers from students' average results for individual types of tasks. If students' success in newly added assignments is significantly higher or significantly lower than the average of works of the same type and level, the assignment will be replaced or removed from the system.
- Involving advanced students in discussions on tasks in introductory programming courses will be a versatile benefit. First, it relieves teachers of the tedious work of answering elementary questions and allows them to tackle more complex tasks. It will bring advanced students new experience from working with less experienced colleagues and ensure their communication skills and patience. Simultaneously, advanced students will learn to accept criticism in case of inconsistent or inaccurate answers. We also anticipate developing relationships between groups of students, which may be used later in study or work for team building. Finally, the benefit for novices will be to get the answer to the question

faster, often in a language that is closer to that of the students' generation than to the teachers' generation.

The final part of the framework is focused on mastering the development environments used by employers and building soft skills in general.

- The most challenging task in the advanced phases defined by the framework is to create new types of activities in the system. A prerequisite for implementing new tasks is a mastery of the technologies by which the system is built. Therefore, students will not create new activities at the beginning of specialised courses but later -- after completing a set of school tasks and at least one complex project. It is assumed that students who complete their education in an educational environment and create content for younger colleagues know the system appropriately. Knowledge of the system functionalities is the first condition for the possibility of its development. In parallel, the possibility of using students' creativity in designing completely new types of activities will be utilised not only for programming but also for other areas forming IT professionals' skills and knowledge (mathematics, artificial intelligence, etc.).
- Developing new applications or upgrading applications to new, more modern environments that will gain a foothold in the application development market in the future is a complex task that requires the involvement of development teams. As part of IT specialists training, courses devoted to team cooperation and communication or leadership skills are usually part of the study. These courses content will be updated and extended by tasks supporting the implemented system's upgrade and development. The tasks will be focused on advancing the existing functionalities to a newer environment or building partial applications using the deployed system's backend infrastructure (e.g., C language lessons, 30 days with Java language). Mobile, web or desktop applications can be created – communication with the backend via the API interface will enable any technology communicating via the HTTP protocol. An alternative design of mobile and web applications has great potential in educational activities in developing new types of activities. Students gain knowledge and practical skills in developing applications in the real world with immediate feedback from users.

Implementing the model and the system described creates a space for further research and verification of many educational theories focused on and verified only within the isolated teaching programming problems. It can be assumed that successful implementation will increase the quality of training of IT specialists.

One of the most important educational system goals is identifying problem students and the early detection of the risk of early course (or study) termination. Therefore, a goal in the near future is to implement algorithms that can detect this group of students and then implement functions and modules that will allow them to overcome the unfavourable situation.

Integration with other education systems and collecting data through other education systems to gain a more accurate and detailed view of the student have been developed and described in (Drlík et al., 2017; Skalka et al., 2020b).

Some ideas for future research based on natural language processing (NLP) focused on automation and artificial intelligence functions have been published in (Skalka, 2018). A valuable technique for preparing new lessons from complex content (e.g., book, articles) is a summary that analyses the content and chooses essential information. The summary techniques can extract whole sentences or unit information from the text, which will become the basis of microlessons or questions. Elements of NLP will create coherent sentences, enabling the generation of meaningful content.

Another logical direction from the collected content is feedback generation for program errors (syntactic and semantic). It is possible to categorise the mistakes and identify the reasons for errors using machine learning methods (Keuning et al., 2018). The data for categorisation is obtained from the submitted correct and incorrect source codes. Submitted source code with errors can be classified, and the system will guide students in correcting the code.

Another exciting element is implementing question-answering methods enabling answer generation based on the educational content, via, for example, questions posted to the student's discussion.

Integrating these ideas requires developing additional software modules based on artificial intelligence tools and prepared and optimised content.

Acknowledgements This research was funded by European Commission under the ERASMUS+ Programme 2018, KA2, grant number: 2018-1-SK01-KA203-046382 "Work-Based Learning in Future IT Professionals Education", Ministry of Education of Slovakia, grant number 004UKF-2-1/2021 "Preparation and development of teaching courses in English with a focus on artificial intelligence in the form of blended-learning", and Ministry of Education of Slovakia, grant number: 2020/8148:34-A1101 "Support for the development of practical skills of UKF students in Nitra".

References

Ala-Mutka, K. M. (2005). A survey of automated assessment approaches for programming assignments. *Computer Science Education, 15*(2). https://doi.org/10.1080/08993400500150747

Baldwin, S. J., & Ching, Y. H. (2020). Guidelines for designing online courses for mobile devices. *TechTrends, 64*(3). https://doi.org/10.1007/s11528-019-00463-6

Bartolomé, A., Castañeda, L., & Adell, J. (2018). Personalisation in educational technology: The absence of underlying pedagogies. *International Journal of Educational Technology in Higher Education, 15*(1). https://doi.org/10.1186/s41239-018-0095-0

Blaženska-Tabakovska, N., Ivanović, M., Klašnja-Milićević, A., & Ivković, J. (2017). Comparison of E-learning personalization systems: Protus and PLeMSys. *International Journal of Emerging Technologies in Learning, 12*(1). https://doi.org/10.3991/ijet.v12i01.6085

Brosig, F., Huber, N., & Kounev, S. (2014). Architecture-level software performance abstractions for online performance prediction. *Science of Computer Programming, 90*. https://doi.org/10.1016/j.scico.2013.06.004

Brusilovsky, P., Malmi, L., Hosseini, R., Guerra, J., Sirkiä, T., & Pollari-Malmi, K. (2018). An integrated practice system for learning programming in python: Design and evaluation.

Research and Practice in Technology Enhanced Learning, 13(1). https://doi.org/10.1186/s41039-018-0085-9

Buffardi, K., & Edwards, S. H. (2014). Introducing CodeWorkout. https://doi.org/10.1145/2538862.2544317.

Carlon, M. K. J., Keerativoranan, N., & Cross, J. S. (2020). Content type distribution and readability of MOOCs. In *L@S 2020 – Proceedings of the 7th ACM conference on learning @ Scale*. Retrieved from https://doi.org/10.1145/3386527.3405950.

Çetin, M., & Demircan, H. Ö. (2020). Empowering technology and engineering for STEM education through programming robots: A systematic literature review. *Early Child Development and Care*. https://doi.org/10.1080/03004430.2018.1534844

Chauhan, A. (2014). Massive Open Online Courses (MOOCS): Emerging trends in assessment and accreditation. *Digital Education Review*. https://doi.org/10.1344/der.2014.25.7-17

Chen, H. M., Chen, W. H., & Lee, C. C. (2018). An automated assessment system for analysis of coding convention violations in Java programming assignments*. *Journal of Information Science and Engineering, 34*(5). https://doi.org/10.6688/JISE.201809_34(5).0006

Crow, T., Luxton-Reilly, A., & Wuensche, B. (2018). Intelligent tutoring systems for programming education: A systematic review. In *ACM International conference proceeding series*. Retrieved from https://doi.org/10.1145/3160489.3160492.

Drlik, M., & Munk, M. (2019). Understanding time-based trends in stakeholders' choice of learning activity type using predictive models. *IEEE. Access, 7*. https://doi.org/10.1109/ACCESS.2018.2887057

Drlík, M., Švec, P., Kapusta, J., Munk, M., Noskova, T., Pavlova, T., et al. (2017). Identification of differences in university e-environment between selected EU and non-EU countries using knowledge mining methods: Project IRNet case study. *International Journal of Web Based Communities, 13*(2). https://doi.org/10.1504/IJWBC.2017.084416

Fernández Alemán, J. L. (2011). Automated assessment in a programming tools course. *IEEE Transactions on Education, 54*(4). https://doi.org/10.1109/TE.2010.2098442

Ferretti, L., Marchetti, M., & Colajanni, M. (2017). Verifiable delegated authorization for user-centric architectures and an OAuth2 implementation. In *Proceedings – International computer software and applications conference* (Vol. 2). Retrieved from https://doi.org/10.1109/COMPSAC.2017.260.

Halvoník, D., & Kapusta, J. (2019). Identifying problematic e-courses content based on students behaviour. In *Lecture notes in electrical engineering* (Vol. 489). Retrieved from https://doi.org/10.1007/978-3-319-75605-9_27.

Halvoník, D., & Kapusta, J. (2020). Framework for E-learning materials optimization. *International Journal of Emerging Technologies in Learning, 15*(11). https://doi.org/10.3991/IJET.V15I11.12721

Hug, T. (2005). Microlearning: A new pedagogical challenge. In *Proceedings of microlearning conference 2005*.

Johnston, T. (2015). Lessons from Moocs: Video lectures and peer assessment. *Academy of Educational Leadership Journal, 19*(2).

Kabathova, J., & Drlik, M. (2021). Towards predicting student's dropout in university courses using different machine learning techniques. *Applied Sciences (Switzerland), 11*(7). https://doi.org/10.3390/app11073130

Keuning, H., Jeuring, J., & Heeren, B. (2018). A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education, 19*(1). https://doi.org/10.1145/3231711

Liebenberg, M., & Jarke, M. (2020). Information systems engineering with digital shadows: Concept and case studies: An exploratory paper. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 12127 LNCS). Retrieved from https://doi.org/10.1007/978-3-030-49435-3_5.

Moon, J., Do, J., Lee, D., & Choi, G. W. (2020). A conceptual framework for teaching computational thinking in personalized OERs. *Smart Learning Environments, 7*(1). https://doi.org/10.1186/s40561-019-0108-z

Morze, N., Varchenko-Trotsenko, L., Terletska, T., & Smyrnova-Trybulska, E. (2021). Implementation of adaptive learning at higher education institutions by means of Moodle LMS. In *Journal of physics: Conference series* (Vol. 1840). Retrieved from https://doi.org/10.1088/1742-6596/1840/1/012062.

Rodríguez-del-Pino, J. C., Rubio-Royo, E., & Hernández-Figueroa, Z. (2012). A virtual programming lab for moodle with automatic assessment and anti-plagiarism features. *Conference on E-Learning, E-Business, entreprise information systems, & E-Government*.

Sharma, R., Banati, H., & Bedi, P. (2012). Adaptive content sequencing for E-learning courses using ant colony optimization. In *Advances in intelligent and soft computing* (Vol. 131 AISC). Retrieved from https://doi.org/10.1007/978-81-322-0491-6_53.

Silva, T. R., Hak, J. L., & Winckler, M. (2017). A behavior-based ontology for supporting automated assessment of interactive systems. In *Proceedings – IEEE 11th international conference on semantic computing, ICSC 2017*. Retrieved from https://doi.org/10.1109/ICSC.2017.73.

Skalka, J. (2018). Data processing methods in the development of the microlearning-based framework for teaching programming languages. *Divai 2018: 12th international scientific conference on distance learning in applied informatics*. Retrieved from https://publons.com/publon/18895954.

Skalka, J., & Drlik, M. (2018a). Conceptual framework of microlearning-based training mobile application for improving programming skills. *Advances in Intelligent systems and computing* (Vol. 725). Retrieved from https://doi.org/10.1007/978-3-319-75175-7_22.

Skalka, J., & Drlik, M. (2018b). Priscilla – Proposal of system architecture for programming learning and teaching environment. *IEEE international conference on application of information and communication technologies*. Retrieved from https://publons.com/publon/27387754.

Skalka, J., & Drlik, M. (2020). Automated assessment and microlearning units as predictors of at-risk students and students' outcomes in the introductory programming courses. *Applied Sciences (Switzerland), 10*(13). https://doi.org/10.3390/app10134566

Skalka, J., Drlik, M., & Obonya, J. (2019). Automated assessment in learning and teaching programming languages using virtual learning environment. *Proceedings of ieee global engineering education conference (EDUCON2017)*. Retrieved from https://doi.org/10.1109/EDUCON.2019.8725127.

Skalka, J., Benko, Ľ., Boryczka, M., Landa, J., & Rodríguez-del-Pino, J. C. (2020a). *Java fundamental*. Retrieved from https://doi.org/10.17846/2020-java1.

Skalka, J., Drlik, M., Obonya, J., & Capay, M. (2020b). Architecture proposal for micro-learning application for learning and teaching programming courses. In *IEEE global engineering education conference, EDUCON* (Vol. 2020–April). Retrieved from https://doi.org/10.1109/EDUCON45650.2020.9125407.

Skalka, J., Drlik, M., Benko, L., Kapusta, J., Del Pino, J. C. R., Smyrnova-Trybulska, E., et al. (2021). Conceptual framework for programming skills development based on microlearning and automated source code evaluation in virtual learning environment. *Sustainability (Switzerland), 13*(6). https://doi.org/10.3390/su13063293

Smyrnova-Trybulska, E., Morze, N., Kommers, P., Zuziak, W., & Gladun, M. (2017). Selected aspects and conditions of the use of robots in STEM education for young learners as viewed by teachers and students. *Interactive Technology and Smart Education, 14*(4). https://doi.org/10.1108/ITSE-04-2017-0024

Vesin, B., Mangaroska, K., & Giannakos, M. (2018). Learning in smart environments: User-centered design and analytics of an adaptive learning system. *Smart learning*. *Environments, 5*(1). https://doi.org/10.1186/s40561-018-0071-0