VIRTUAL LEARNING ENVIRONMENT FOR PROGRAMMING LEARNING

Ján Skalka, SK

Abstract: The ability to prepare algorithms to solve problems and rewrite them into program codes is one of the necessary skills for finding work in the IT sector. Although pupils have been developing algorithmic skills since the first years of primary school, many fail to establish it to write programs. In higher education programming, language courses are still the most challenging courses for which students fail. The current approach to solving this problem is based on adapting the educational methodology to the habits of current students. The approach used to obtain information for young Generation Z programmers is based primarily on using smartphones, the limited amount of information displayed, and providing immediate feedback. An ideal tool for teaching programming at universities, which can eliminate some beginner's difficulties when set up correctly, is automated assessment supported by microlearning lessons and the possibility of using generative AI to explain frequently repeated questions. The lecture aims to present the possibilities currently offered by virtual educational environments and to share experiences with their use.

Keywords: programming language, virtual learning environment, programming learning, programming teaching, microlearning.

1. Introduction

The demand for IT professionals in the labor market has increased significantly in recent years [1]. However, the current education system struggles to meet these demands effectively. Traditional IT education at universities, which often relies on a combination of lectures and laboratory exercises, is increasingly considered outdated and ineffective. This approach raises concerns about the effectiveness of students' knowledge acquisition and skill development in IT disciplines [2].

Modern students show a decreasing interest in passive activities such as attending lengthy lectures. Instead, they prefer to apply the acquired knowledge and skills immediately. They value flexibility, looking for learning opportunities that can be accessed anytime, anywhere, rather than

being limited to a school environment [3]. To address these shifts, it is essential to recognize that today's youth are generally more dynamic than previous generations, moving toward skills and knowledge with clear, short-term benefits and tangible applications.
Writing source code has become a core skill in today's workforce. Various support systems have been developed to help teach programming, varying in scope, content, and quality [4]. Many researchers are now looking to expand their focus beyond programming, examining modern learning environments in a broader context, often exploring their intersection with STEM/STEAM fields [5].
The article aims to present architecture, current state, and experience with the virtual learning environment named Priscilla developed by an international team in three following Erasmus projects [6]. The system is based on the conceptual framework for teaching and learning programming [7].
This environment effectively integrates modern, forward-looking educational approaches such as microlearning [8] and automated source code evaluation (automated assessment) [9], [10]. The balanced combination of these methods enables effective teaching time management, allowing students to quickly apply newly acquired knowledge, minimize time spent evaluating source code, and receive immediate feedback – an essential component for mastering programming.

2. Programming Teaching

The long-term goal of programming education is to equip students with the knowledge and skills to solve real-world problems using appropriate programming languages. Problem-solving is a core component of technology courses, and while it occasionally appears in information processing courses, its integration is not always practical or well-executed [11]. Research has shown that beginners tend to rush through problems, spending minimal time interpreting the problem, while experts approach the problem through iterative refinement and gradual understanding [12].
To address these challenges, we propose a two-tiered educational model for teaching programming structured as follows:

- Introductory programming (IP) courses are designed to develop computational thinking and establish a basic understanding of programming languages. They aim to provide a broad knowledge base that is applicable across a variety of computer science disciplines. The content of these

courses is designed to be versatile and broadly applicable to various computer science programs.
- Technology courses focus on specific technologies like web development, server-side programming, mobile application development, databases, and the Internet of Things. They are designed to teach students how to apply technologies in a practical environment and equip them with the skills to solve real-world problems using these tools. These courses emphasize hands-on experience and problem-solving with current technologies.

IP courses should be designed with a clear focus on the diverse learning paths of students, in line with the ARCS model of motivation, which emphasizes attention, relevance, confidence, and satisfaction [13]. Motivation, engagement, and satisfaction are critical in preventing learning failure and ensuring student success. To achieve goals, IP courses should incorporate interactive assignments, immediate feedback, and gamification elements – key components of the ARCS model. These features help sustain student interest and provide tangible incentives, such as earning badges and achieving high rankings, reinforcing motivation, and encouraging continued learning.
Our proposed model for teaching programming is based on the microlearning principles that integrate interactive microlearning lessons alongside automated source code assessment to support IP courses. These characteristics are involved via interactive assignments, immediate feedback, and gamification elements, creating an evitable part of the model. They support achieving the objectives, gaining badges, and placing in the rankings [14].

2.1. Microlearning

Microlearning is a relatively new concept that has gained attention in various fields of learning, didactics, and education. It is characterized by short learning efforts that require minimal time commitment, focusing on small, specific content units and narrow topics [15]. Microlearning offers an innovative approach to organizing and designing learning experiences, emphasizing small, digestible steps where students actively construct their understanding through structured content [16].
This approach delivers content in manageable chunks, making it easier for students to absorb and apply the material quickly. This aligns well with the preferences of today's students, who often seek flexible, concise learning experiences. In addition, microlearning increases learning efficiency by

reducing cognitive overload, allowing students to focus on a single concept without feeling overwhelmed.

In our model, microlearning activities are designed to actively engage students through concise lessons and focused content that address key programming concepts. These lessons focus on specific programming techniques and concepts, allowing students to practice and reinforce their understanding immediately. The content is structured to build progressively, allowing students to progress at their own pace while ensuring continuous reinforcement of knowledge.

2.2 Automated assessment

Teaching programming is a complex educational process that requires a multifaceted approach. While educational theories and taxonomies are valuable tools for developing educational objectives and assessing student achievement, they are not directly applicable to teaching programming. According to [17], programming requires an understanding of the relevant theory and the ability to apply this knowledge to solve real-world problems.

Researchers have proposed different approaches to teaching programming based on different pedagogical theories, didactic frameworks, and educational strategies. Some have sought to simplify the complexity of acquiring knowledge and skills, while others have mapped the process with greater precision. For example, microlearning has been identified as a helpful tool in engineering education, especially for introducing topics or reinforcing concepts. However, microlearning is best suited for conveying only a portion of the knowledge and skills needed for programming. More complex areas of programming, such as application modeling or the use of specialized languages (e.g., SQL), require a different, more comprehensive approach.

In engineering education, programming remains a key component, with key activities such as writing code, testing, and debugging playing a central role. In recent years, the role of the human teacher has evolved, with an increasing reliance on automated evaluation of programming code to support student learning.

Automated assessment tools have become an integral part of programming education, with many systems now used to assess students' coding skills. Some researchers, such as those in [20], have designed systems to grade short text responses in programming tasks automatically. In contrast, others [19] have focused on automated assessments for specific languages, such as C, showing correlations between hands-on exercise completion and overall course performance. In [21], comparing results from introductory Java

programming courses over three years revealed significant improvements in student performance using automated assessments.

These automated assessment tools are often available as Learning Management System (LMS) plugins, such as VPL for Moodle [22], or through independent web portals such as hackerrank.com, codewars.com, and freecodecamp.com. While LMS plugins are commonly used in university settings, independent portals are often aimed at helping individuals demonstrate their programming skills for potential employment opportunities.

Despite the advantages of these solutions, many focus on a single programming language and may involve detailed didactic elaboration [23]. The content used to create assignments for automated assessments must first be carefully prepared and didactically revised [24]. Once this foundation is in place, the content can be transformed into a sequence of tasks associated with microlearning lessons.

However, it is essential to note that simply implementing automated code evaluation tools without thoughtful integration into the overall learning process may not significantly improve learning outcomes.

3 Framework Proposal

Successful and sustainable implementation of the framework requires a comprehensive approach, including integrating introductory programming courses and activities to develop future learning environments and content. To ensure continuous improvement, the framework must include regular updates and new content creation while adhering to modern design trends. These tasks can be addressed through advanced engineering courses, where students can actively contribute to developing the learning environment they are familiar with, having studied programming.

The implementation of the framework, as outlined in [1], started by a clear concept and learning processes implemented in the LMS Moodle [7] and continued into the newly developed system Priscilla [6]. In this setting, tests using different quiz questions were designed to meet the needs of microlearning. These tests include simple answer formats, such as selecting options to complete the source code.

Automated source code evaluation was facilitated through a virtual programming laboratory that supports automatic code evaluation in multiple programming languages [18].

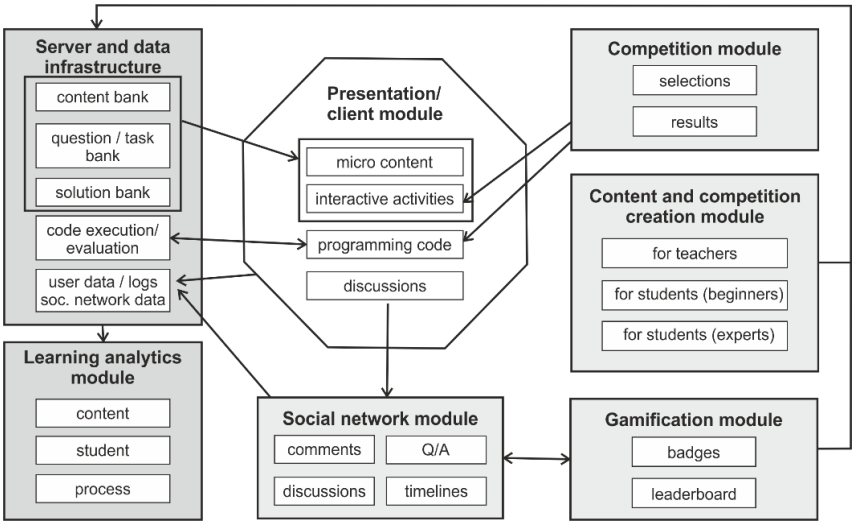

Fig. 1 Simplified framework structure [19]

The structure of the framework is designed to focus on simplicity, extensibility, and modularity, creating complex learning structures from essential components. At its core, the system is built on three basic microlearning units that serve as building blocks for more advanced learning features.

- A content activity is a basic unit providing learning material in formats like HTML documents, images, or videos. This unit provides students with background information and facilitates discussions about the content to improve understanding.
- An interactive activity adds a layer of engagement by presenting questions, tasks, or assignments that require active student participation.
- A code activity allows students to write and submit programming code, the correctness of which is then evaluated by a server, which requires an online connection.

These basic units are combined to create more complex structures. A lesson sequences content and interactive activities provide a cohesive learning experience, while a quiz focuses on interactive tasks that serve as a tool for review, feedback, or validation of skills. Students can use quizzes to skip lessons if they already have the required knowledge.

The system also includes competitions, which emphasize accuracy and speed through algorithmically generated interactive tasks and coding activities, where students write, and test source code based on unlocked tasks upon lessons completion.
These logical units act as containers that allow for integrating different activities, including those introduced in later stages of development.

4 Practical implementation

The unique nature of learning programming languages requires immediate interaction or feedback from instructors or peers when students encounter problems. Providing immediate feedback is especially crucial for novice programmers, as its absence can lead to a misunderstanding of basic programming concepts [16], [17]. This feedback can be provided by a human instructor or through automatically generated responses integrated into various frameworks and platforms designed for teaching programming teaching [12], [18], [19]. An educational system must cover all phases and aspects of learning programming to be effective.
The PRISCILLA system, shown in Figure 2, is designed with a modular architecture comprising independent front-end (presentation/client module) and backend components that communicate via an API. The API facilitates data exchange in JSON format, chosen for its lightweight and faster performance compared to XML-RPC, while meeting all standard requirements [34]. This structure ensures flexibility, scalability, and efficient communication between system components.
Thanks to this approach, even though the basis of the system is defined in the form of a web application, several mobile applications (in the form of student theses) have been created. These applications use the system features and data and enable learning within selected courses independently of the standard front-end interface.

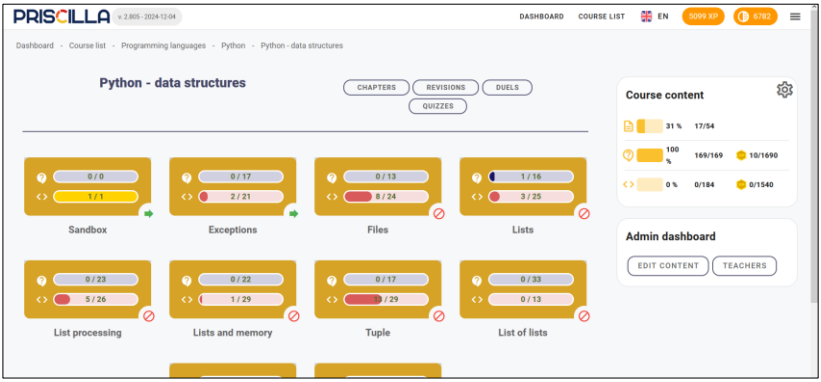

Fig. 2 View of the current content of PRISCILLA

The system is designed to support teaching multiple programming languages within a single application, offering flexibility and scalability for different learning needs. Each programming language is associated with a predefined default lesson plan (Figure 2), which is carefully structured to include micro-learning activities (focused on learning theoretical concepts) and programming assignments (focused on practical coding tasks).

Students' progress through the lesson plan sequentially; however, the system support reorders chapters or lessons in the content and activities based on individual performance and preferences. While ensuring that no critical activities aimed at acquiring essential knowledge and skills are skipped, the student can select the order of lessons to align with his current knowledge level, learning pace, and preferences.

This personalized approach accommodates diverse learning paths and increases engagement by tailoring the learning experience to each individual. By balancing structured lesson plans with flexibility in the sequence of activities, the system supports mastery of core skills and deeper exploration of programming concepts, ensuring a more effective and student-centred learning process.

4. Learning Forms Suitable for Teaching AI

University students majoring in IT often expect effective learning methods that balance acquiring theoretical knowledge with developing practical skills, all with an emphasis on simplicity and relevance. This generation seeks to acquire specialized knowledge and skills in areas such as AI in a way that aligns with their habits and career aspirations – regardless of whether they

ultimately pursue a career in AI or not. To address this challenge, a work-based learning strategy that integrates elements of active learning, collaborative learning, and problem-based learning to increase student engagement and prepare them for AI-related careers has been adopted.

Active learning is central to this approach because it places responsibility for educational progress on the students themselves. This method encourages engagement through various forms of activation, including increased interaction, collaboration, a more profound exploration of the material, and critical thinking. Research [20] highlights the effectiveness of active learning and demonstrates its superiority over passive teaching methods. In this regard, active learning emphasizes student-led inquiry and responsibility, which is particularly important for mastering AI concepts and skills. Also other authors [21] proved that active learning leads to better outcomes than comparatively passive forms of instruction.

Suppose studying AI is like the cognitive leap required to learn programming. The benefits of flipped classroom models in programming learning should also be considered. Flipped classrooms have been shown to improve student achievement [22], [23], but their success depends on strict adherence to principles that ensure that students engage with the learning materials even outside of class [24].

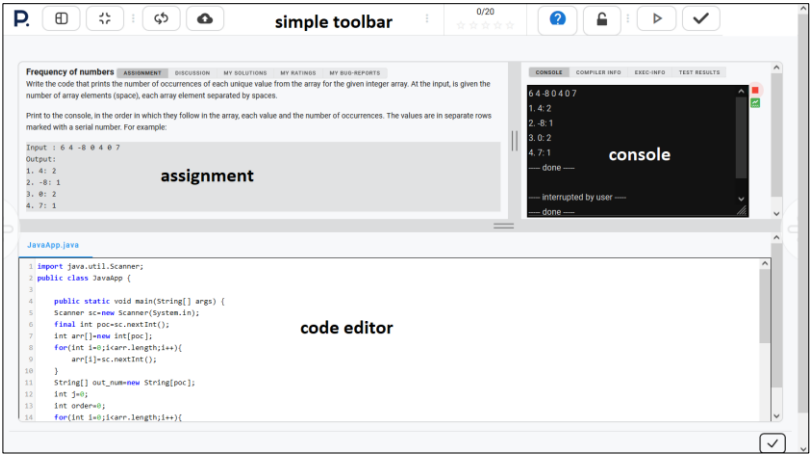

Fig. 3 Web interface dedicated to solving and code writing in the PRISCILLA system [25]

Building on these learning strategies, the FITPED and FITPED-AI consortium – involving universities and small and medium-sized enterprise organizations – extended and optimized a learning model originally designed for

programming languages [1], [6] to include AI learning. A first change of the original version of the system was adaptation of the web-based development environment to allow students to write, run, and debug programs directly within the platform in one web page. This efficient approach saves time and effort and allows students to focus on problem-solving and experimentation. The system runs code on the server, ensuring a seamless experience for users while supporting the collaborative and interactive aspects necessary for active learning (Figure 3).

Leaving the task solution to powerful server processors is a standard approach, even in environments focused on solving AI tasks. The primary reason is that, in many cases, programs need high performance and a long time to achieve the result, which cannot generally be provided on local devices.

The requirement for solving tasks and experimenting with data is currently most often implemented by the Jupyter Notebook technology [26]. Due to its openness, simplicity, and continuous development, it has become a popular tool in data science and AI teams. It is currently used as a format used for data processing in science and education [27]. Its strength lies in combining text and source code and editing and running this code at any time with a single click. In addition, the results are or can be displayed as part of the document content.

The Jupyter server/notebook technology has a significant disadvantage, which the authors recently identified during its maturation – to use the computational and processing components, it is necessary to run content from a given server – notebooks could operate via localhost by default [28]. This approach made working with other systems and front-end applications difficult or impossible.

The Jupyter Kernel Gateway (JKG) technology is currently used as one of the alternatives to enable communication between an independent front-end and a Jupyter server running on the backend. According to [29], JKG is a web server that provides headless access to Jupyter kernels. As a result, independent applications communicate with the kernels remotely via REST calls and web sockets instead of ZeroMQ messages [30].

JKG made it possible to implement modules that provided communication with Python kernels, which are commonly used for solving data science and artificial intelligence tasks. One kernel can be connected to one or more front-ends simultaneously.

To integrate the Jupyter infrastructure into the Priscilla system in use and enable communication with Python kernels, it was necessary to create a clone of the standard Jupyter notebook design and enrich it with possible additional functions (the ability to stop the program, friendly insertion of input data into

a running program, the ability to combine with rich text, etc.). An example of the prepared content (from the FITPED-AI project) is shown in Figure 4.

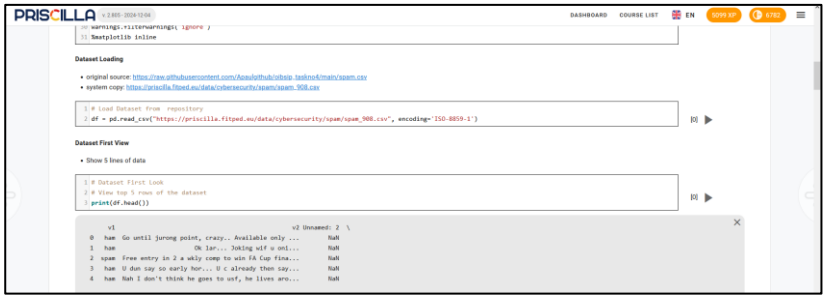

Fig. 4 Integration of the Jupyter Notebook design into the Priscilla system

Newly developed modules needed to teach AI courses will provide immediate feedback and support students' projects in AI. The created educational content will consist of lessons for learning prerequisites of AI, classes for teaching basics of AI (data preparation, knowledge discovery, artificial intelligence, machine learning), and courses for teaching application domains of AI (natural language processing, educational data, cybersecurity). In addition, educational data will be collected within several rounds of courses, which will be used to identify students' behavior and problem areas in the educational content and teaching process.

5 Conclusion

A key consideration in designing course architecture is determining whether students should have the freedom to navigate content independently or follow a structured, directive path. Allowing free navigation is generally well-received by students, enabling them to select chapters based on their immediate needs, regardless of sequence.

Feedback from students and the outlined observations highlight the potential to define and generalize principles for creating programming courses. These principles leverage a microlearning approach to introduce and reinforce content, utilizing automated assessment for hands-on programming practice and skill development.

The following principles for content creation have been established to optimize learning outcomes and enhance student engagement [31].

- Course design
  - Allow students to freely navigate course content in any order to encourage personalized learning and maintain motivation.

- Include enough detailed tasks and microlearning tasks to support weaker students, while keeping tasks varied to engage different skill levels.
- Focus on teaching programming through hands-on practice, avoiding unnecessary delays or boredom.

- Microlearning activities
  - Keep lessons concise and interspersed with microquestions to immediately reinforce learning.
  - Keep questions short, clear, and connected to programming concepts, gradually increasing in complexity.
  - Avoid repetitive microcontent and ensure tasks are equally scored to encourage completion without bias.
  - Use progressive programming exercises, from command selection to independent coding.
- Automated assessment
  - Design programming tasks that are clear, concise, and aligned with learning objectives, with increasing complexity.
  - Use algorithmic test cases to challenge students and prevent simple guessing.
  - Assign task scores based on difficulty and optimize system resources by considering computational complexity.

The advent of Generative AI has significantly transforms the teaching landscape, introducing innovative tools and methodologies that redefine how education is delivered and experienced. Educators now have access to AI-driven platforms capable of creating customized lesson plans, generating interactive content, and providing real-time feedback to students. This shift has enhanced the efficiency of teaching and empowered teachers to focus more on fostering critical thinking and problem-solving skills. With AI tools capable of automating routine tasks such as grading or administrative work, educators can dedicate more time to individualized instruction and mentoring, addressing diverse student needs more effectively.

Generative AI has also reshaped how students interact with educational material, enabling a more personalized and adaptive learning experience. AI systems can analyze student performance and preferences, dynamically adjusting the difficulty level and content delivery style to suit each learner. Tools like AI-generated practice questions, virtual tutors, and creative brainstorming aids have made learning more engaging and accessible.

Understanding Generative Artificial Intelligence (GAI), including its strengths and weaknesses, ethical considerations, and trustworthiness, is

becoming essential in many professions. The following project of the FITPED consortium – FITPED GAI – aims to design and validate an educational model that addresses these needs by integrating GAI tools into the educational process. The concept will cover the needs of students during learning, help teachers prepare for the lesson, and equip future employees for workplace applications. This will be achieved through structured courses, methodologies, and workshops.
The main output will be a validated educational model consisting of online courses focused on AI and GAI while aligning with innovations in the education system. It emphasizes capacity building for areas beyond IT, training teachers in GAI integration, and developing advanced programming and innovation skills.

This work has been supported by the European Commission under the ERASMUS+ Programme 2021, KA2 under Grant 2021-1-SK01-KA220-HED-000032095: Future IT Professionals Education in Artificial Intelligence and KA220-HED under Grant 2024-1-SK01-KA220-HED-000249044: Future IT Professional Education in Generative Artificial Intelligence.

References

[1] J. Skalka and M. Drlík, *Conceptual framework of microlearning-based training mobile application for improving programming skills*, vol. 725. 2018.

[2] G. M. M. Bashir and A. S. M. L. Hoque, 'An effective learning and teaching model for programming languages', *Journal of Computers in Education*, vol. 3, no. 4, 2016, doi: 10.1007/s40692-016-0073-2.

[3] A. Granić, 'Educational Technology Adoption: A systematic review', *Education and Information Technologies*, vol. 27, no. 7, 2022, doi: 10.1007/s10639-022-10951-7.

[4] T. Crow, A. Luxton-Reilly, and B. Wuensche, 'Intelligent Tutoring Systems for Programming Education: A Systematic Review', 2018, doi: 10.1145/3160489.3160492.

[5] M. Çetin and H. Ö. Demircan, 'Empowering technology and engineering for STEM education through programming robots: a systematic literature review', *Early Child Development and Care*, vol. 190, no. 9. 2020, doi: 10.1080/03004430.2018.1534844.

[6] J. Skalka and M. Drlik, 'Priscilla - Proposal of System Architecture for Programming Learning and Teaching Environment', *IEEE International Conference on Application of Information and Communication Technologies*, 2018, [Online]. Available: https://publons.com/publon/27387754/.

[7] J. Skalka *et al.*, 'Conceptual framework for programming skills development based on microlearning and automated source code evaluation in virtual learning environment', *Sustainability (Switzerland)*, vol. 13, no. 6, 2021, doi: 10.3390/su13063293.

[8] T. Hug, 'Microlearning : A New Pedagogical Challenge', 2005.

[9] K. M. Ala-Mutka, 'A survey of automated assessment approaches for programming assignments', *Computer Science Education*, vol. 15, no. 2, 2005, doi: 10.1080/08993400500150747.

[10] J. L. Fernández Alemán, 'Automated assessment in a programming tools course', *IEEE Transactions on Education*, vol. 54, no. 4, 2011, doi: 10.1109/TE.2010.2098442.

[11] A. P. Ambrósio, F. M. Costa, L. Almeida, A. Franco, and J. Macedo, 'Identifying cognitive abilities to improve CS1 outcome', 2011, doi: 10.1109/FIE.2011.6142824.

[12] A. Gomes and A. J. Mendes, 'An environment to improve programming education', in *ACM International Conference Proceeding Series*, 2007, vol. 285, doi: 10.1145/1330598.1330691.

[13] S. Alhazbi, 'ARCS-based tactics to improve students' motivation in computer programming course', 2015, doi: 10.1109/ICCSE.2015.7250263.

[14] J. Skalka, 'Microlearning and Automated Assessment -- A Framework Implementation of Dissimilar Elements to Achieve Better Educational Outcomes', in *Microlearning: New Approaches To A More Effective Higher Education*, E. Smyrnova-Trybulska, P. Kommers, M. Drlík, and J. Skalka, Eds. Cham: Springer International Publishing, 2022, pp. 1–26.

[15] T. Hug and N. Friesen, 'Outline of a Microlearning agenda', *... of Microlearning. Concepts, Discourses and ...*, no. September, 2007.

[16] L. F. Zulueta and J. F. D. Panoy, 'Scenario-Based Microlearning Strategy for Improved Basic Science Process Skills in Self-Directed Learning', *International Journal of Science, Technology, Engineering and Mathematics*, vol. 2, no. 4, 2022, doi: 10.53378/352932.

[17] H. C. Looi and A. H. Seyal, 'Problem-based Learning: An Analysis of its Application to the Teaching of Programming', in *International Proceedings of Economics Development and Research*, 2014, vol. 70, no. 14.

[18] J. C. Rodríguez-del-Pino, E. Rubio-Royo, and Z. Hernández-Figueroa, 'A Virtual Programming Lab for Moodle with automatic assessment and anti-plagiarism features', *Conference on e-Learning, e-Business, Entreprise Information Systems, & e-Government*, 2012.

[19] J. Skalka and M. Drlík, 'Educational Model for Improving Programming Skills Based on Conceptual Microlearning Framework BT - The Challenges of the Digital Transformation in Education', 2020, pp. 923–

934.

[20] S. Hartikainen, H. Rintala, L. Pylväs, and P. Nokelainen, 'The concept of active learning and the measurement of learning outcomes: A review of research in engineering higher education', *Education Sciences*, vol. 9, no. 4, 2019, doi: 10.3390/educsci9040276.

[21] D. B. Markant, A. Ruggeri, T. M. Gureckis, and F. Xu, 'Enhanced Memory as a Common Effect of Active Learning', *Mind, Brain, and Education*, vol. 10, no. 3. 2016, doi: 10.1111/mbe.12117.

[22] H. Özyurt and Ö. Özyurt, 'Analyzing the effects of adapted flipped classroom approach on computer programming success, attitude toward programming, and programming self-efficacy', *Computer Applications in Engineering Education*, vol. 26, no. 6, 2018, doi: 10.1002/cae.21973.

[23] M. K. P. Peethambaran, V. G. Renumol, and S. Murthy, 'Flipped Classroom Strategy to Help Underachievers in Java Programming', 2018, doi: 10.1109/LaTICE.2018.000-7.

[24] J. Skalka and M. Drlik, 'Automated assessment and microlearning units as predictors of at-risk students and students' outcomes in the introductory programming courses', *Applied Sciences (Switzerland)*, vol. 10, no. 13, 2020, doi: 10.3390/app10134566.

[25] J. Skalka and M. Drlik, 'Proposal of Artificial Intelligence Educational Model Using Active Learning in a Virtual Learning Environment', 2022, pp. 15–28.

[26] F. Pérez and B. E. Granger, 'Project Jupyter : Computational Narratives as the Engine of Collaborative Data Science', *UC Berkeley*, no. April, 2015.

[27] J. W. Johnson, 'Benefits and Pitfalls of Jupyter Notebooks in the Classroom', 2020, doi: 10.1145/3368308.3415397.

[28] Project Jupyter, 'Standalone Jupyter server enhancement', 2022. https://jupyter.org/enhancement-proposals/28-jupyter-server/jupyter-server.html (accessed Aug. 30, 2022).

[29] Project Jupyter Team, 'Jupyter Kernel Gateway', 2022. https://jupyter-kernel-gateway.readthedocs.io/en/latest/ (accessed Aug. 30, 2020).

[30] P. Hintjens, *ZeroMQ: messaging for many applications*. ' O'Reilly Media, Inc.', 2013.

[31] J. Skalka, L. Benko, M. Drlik, M. Munk, and P. Svec, 'Guidance for Introductory Programming Courses Creation Using Microlearning and Automated Assessment', in *Microlearning*, 2022.

Contact address

RNDr. Ján Skalka, PhD.

Faculty of Natural Sciences and Informatics
Constantine the Philosopher University in Nitra
Tr. A. Hlinku 1
949 01 Nitra
e-mail: jskalka@ukf.sk