ARTIFICIAL INTELLIGENCE-BASED ADAPTIVE TEACHING OF PROGRAMMING IN THE DIGITAL EDUCATION ENVIRONMENT

Sulaymonova Dilnoza Bohodir qizi

Assistant Professor, Karshi State Technical University,

Uzbekistan, Karshi

Abstract. The article examines the scientific, theoretical and methodological foundations of artificial intelligence-based adaptive teaching of programming in the digital education environment. The relevance of the study is determined by the heterogeneity of students' initial training, differences in algorithmic thinking, the need for rapid diagnostic feedback and the methodological limitations of traditional learning management systems.

Keywords: artificial intelligence, adaptive teaching, programming education, digital education, learning analytics, individual learning trajectory, automated feedback.

АДАПТИВНОЕ ОБУЧЕНИЕ ПРОГРАММИРОВАНИЮ НА ОСНОВЕ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В ЦИФРОВОЙ ОБРАЗОВАТЕЛЬНОЙ СРЕДЕ

Сулаймонова Дилноза Баходир қизи

Ассистент, Каршинский государственный технический университет

Узбекистан, г. Карши

Аннотация. В статье рассматриваются научные, теоретические и методологические основы адаптивного обучения программированию на основе искусственного интеллекта в цифровой образовательной среде. Актуальность исследования определяется неоднородностью начальной подготовки студентов, различиями в алгоритмическом мышлении, необходимостью быстрой диагностической обратной связи и методологическими ограничениями традиционных систем управления обучением.

Ключевые слова: искусственный интеллект, адаптивное обучение, обучение программированию, цифровое образование, learning analytics, индивидуальная образовательная траектория, автоматизированная обратная связь.

Introduction. The rapid development of digital education requires higher education institutions to reconsider the content, forms and methodological support of teaching. This requirement is especially evident in programming courses, where students enter the

learning process with different levels of prior knowledge, different rates of mastering concepts, and different abilities to interpret algorithms, trace code and detect errors. Therefore, the digitalization of programming education cannot be reduced to posting lectures, tests and assignments on a platform. It should be organized as a flexible pedagogical system in which the learner's current state is continuously diagnosed and the learning process is adapted on the basis of reliable educational data.

Traditional LMS platforms provide organizational support for learning, but their personalization capabilities remain limited. They often record outcomes without revealing the causes of programming difficulties, which highlights the importance of adaptive teaching.

Artificial intelligence technologies open new possibilities for such adaptation. Machine learning models can predict difficulties, learning analytics can reveal activity patterns, knowledge tracing can track the acquisition of specific programming concepts, recommender systems can select relevant resources and generative AI can provide explanatory feedback. At the same time, the use of AI in education requires pedagogical supervision, transparency, data protection and a balance between automation and the development of independent thinking.

Materials and Methods. The study is based on theoretical analysis, comparative generalization, pedagogical modeling and systematic interpretation of contemporary research on adaptive learning, intelligent tutoring systems, learning analytics, educational data mining and generative AI in programming education. The methodological focus is placed on the relationship between digital indicators of learning activity and didactic decisions made by the teacher or by an AI-supported adaptive system.

The source material was transformed from a dissertation chapter devoted to the scientific and methodological foundations of AI-based adaptive teaching in digital education. The chapter content was adapted to the structure of a journal article by narrowing the scope, removing dissertation-style subdivisions, formulating the aim and methods explicitly, and presenting the main results through a compact table and an integrated adaptive mechanism.

Programming education is viewed as the integrated development of conceptual knowledge, practical skills and problem-solving abilities.

Results and Discussion. The analysis shows that the main didactic problem of programming education in a digital environment is the mismatch between a uniform learning path and the diversity of students' learning states. The same topic, the same task and the same assessment criteria do not have the same educational value for all students. A learner who has mastered conditional statements but struggles with loops needs visual algorithmic explanations and gradual practice. A stronger learner may need complex problem-based tasks, refactoring, optimization or project components.

In this context, adaptive teaching can be defined as a dynamic didactic system that adjusts learning content, task complexity, feedback format and repetition frequency according to the learner's current state. The central element of this system is the learner model. In programming education, the model should include not only test scores, but also the level of algorithmic thinking, the pace of task completion, the number of attempts, the types of code errors, the quality of explanation and the response to feedback.

A simplified representation of the learner model can be expressed as follows: $L = \{K, E, P, A, F\}$, where K denotes the knowledge state, E denotes the error profile, P denotes the learning pace, A denotes activity indicators, and F denotes the learner's response to feedback. The adaptive decision can then be represented as $R = f(L, C, G)$, where C is the content model and G is the learning goal. This formulation emphasizes that adaptation is not a random recommendation, but a pedagogically justified decision based on the relationship between the learner's state, the structure of the content and the intended competence.

Table 1.

Pedagogical problems and adaptive AI-based solutions in programming education

Pedagogical problem	Digital indicator	Adaptive methodological solution	AI capability
Different initial preparation	Entry test, diagnostic coding tasks, initial topic score	Grouping by level and selecting a suitable content route	Classification model and knowledge-state assessment
Difficulty in algorithmic thinking	Problem-solving steps, tracing, flowcharts, code comments, number of attempts	Gradually increasing tasks and visual explanations	Recommendation system and analysis of the solution path
Frequent syntactic and semantic errors	Compiler/interpreter messages, test-case results, runtime errors	Micro-practice and explanatory feedback according to the error profile	Automated feedback and error classification
Different learning pace	Task completion time, repeated attempts, delays	Individual pace and repetition intervals	Learning analytics and progress prediction
Weak motivation and	Platform activity, resource use,	Process assessment, reflection	Human-in-the-loop

excessive reliance on AI	similarity of code, lack of explanation	and defense tasks	monitoring and explainable assessment

Table 1 demonstrates that each programming difficulty has a measurable digital indicator and can be connected with an adaptive pedagogical decision. In such a system, artificial intelligence does not replace the teacher; it transforms raw digital traces into diagnostic information and supports the teacher's decision-making. This distinction is essential because the same numerical indicator may have different pedagogical meanings. For example, many attempts may indicate insufficient understanding, but they may also show persistence and independent exploration.

The main didactic functions of AI include diagnostics, prediction, adaptation, feedback and analytical support for teachers.

Generative AI has particular value in programming education because it can provide natural-language explanations of code, generate test cases, propose alternative solutions and answer student questions in dialogue form. Nevertheless, its use must be controlled methodologically. If the learner receives ready-made code without explaining its logic, the visible result may improve while algorithmic thinking remains undeveloped. Therefore, assignments should include explanation, reflection, code defense and process assessment.

Didactic mechanism of AI-based adaptive teaching. The proposed mechanism includes continuous diagnosis, learner-model updating, adaptive recommendation generation, individualized feedback and subsequent pedagogical interpretation by the teacher.

Conclusion. The study confirms that AI-based adaptive teaching of programming is a necessary methodological direction in the digital education environment. Programming courses require personalization because students differ in prior knowledge, algorithmic thinking, pace of learning, error patterns and motivation. A uniform digital course cannot fully respond to these differences.

Traditional LMS platforms provide important infrastructure, but they are not sufficient for dynamic methodological decision-making. Effective adaptive teaching requires a learner model, an error profile, a content model, adaptive tasks and individualized feedback. Artificial intelligence technologies can support these elements through diagnostics, prediction, recommendation, automated feedback and learning analytics.

At the same time, AI should be interpreted as a didactic mechanism that supports the teacher rather than a tool that replaces pedagogical judgment. The effectiveness of adaptive teaching depends on the integration of technological possibilities with learning objectives, formative assessment, academic integrity, transparency and human-centered pedagogical control. These findings provide a theoretical basis for designing an AI-supported adaptive methodology for teaching programming in higher education.

References

1. Brusilovsky P. Adaptive hypermedia // User Modeling and User-Adapted Interaction. – 2001. – Vol. 11. – P. 87–110.
2. Corbett A.T., Anderson J.R. Knowledge tracing: Modeling the acquisition of procedural knowledge // User Modeling and User-Adapted Interaction. – 1995. – Vol. 4. – P. 253–278.
3. Woolf B.P. Building Intelligent Interactive Tutors: Student-centered strategies for revolutionizing e-learning. – Burlington: Morgan Kaufmann, 2009.
4. Koedinger K.R., Corbett A.T., Perfetti C. The Knowledge-Learning-Instruction framework: Bridging the science-practice chasm to enhance robust student learning // Cognitive Science. – 2012. – Vol. 36(5). – P. 757–798.
5. VanLehn K. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems // Educational Psychologist. – 2011. – Vol. 46(4). – P. 197–221.
6. Siemens G. Learning analytics: The emergence of a discipline // American Behavioral Scientist. – 2013. – Vol. 57(10). – P. 1380–1400.
7. Ferguson R. Learning analytics: drivers, developments and challenges // International Journal of Technology Enhanced Learning. – 2012. – Vol. 4(5/6). – P. 304–317.
8. Romero C., Ventura S. Educational data mining and learning analytics: An updated survey // Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. – 2020. – Vol. 10(3).
9. Sweller J., van Merrienboer J.J.G., Paas F. Cognitive architecture and instructional design: 20 years later // Educational Psychology Review. – 2019. – Vol. 31. – P. 261–292.
10. Kodirov, B. (2026). O'QUVCHILARDA KOMPLEKS MA'LUMOTLAR BAZASI KOMPETENSIYALARINI RIVOJLANTIRISH UCHUN TIZIMLANGAN PEDAGOGIK STRATEGIYALAR. *Osiyo ilmiy tadqiqotlar va innovatsiyalar jurnali* , *5* (1), 143-147.
11. Бекматов А.К., & Рустамов Т.С. (2024). РОЛЬ ГЛУБОКОГО ОБУЧЕНИЯ В УЛУЧШЕНИИ ТОЧНОСТИ СИСТЕМ ОБНАРУЖЕНИЯ ВТОРЖЕНИЙ. Экономика и социум, (6-1 (121)), 1582-1591.