AN INTERPRETABLE AI MODEL FOR INTELLIGENT IT SUPPORT TICKET PRIORITIZATION IN DIGITAL ORGANIZATIONS

Buriyeva Gulina Bakhtiyor kizi
University of Information Technologies and Management, student

Abstract: *Programming courses require continuous practice, timely feedback and the ability to understand not only whether a program works, but why it fails. Traditional assessment usually checks final output or syntax errors, while many student difficulties are connected with hidden misconceptions: boundary cases, variable scope, loop logic, data structure choice and algorithmic thinking. This paper proposes an explainable artificial intelligence-assisted framework for adaptive feedback in programming courses. The framework combines code analysis, test execution, learning analytics and natural-language explanation into a transparent feedback cycle. Instead of replacing the teacher, the model supports the teacher by detecting frequent error patterns, estimating the learner misconception level, and recommending the next task according to the student profile. A scenario-based analytical comparison shows that the proposed approach can improve feedback clarity, reduce delayed correction and support individualized learning in programming education.*

Keywords: *artificial intelligence, ICT in education, programming courses, adaptive feedback, code analysis, learning analytics, explainable AI, digital learning.*

INTRODUCTION

Programming is one of the core disciplines in information and communication technology education. It develops algorithmic thinking, problem decomposition, debugging skills and the ability to build digital solutions. However, programming is also difficult for beginners because a small syntactic mistake may hide a deeper conceptual problem. A student may know the definition of a loop but fail to use it correctly in a boundary case; another student may pass simple tests but still write an inefficient or fragile solution.

In many educational settings, teachers cannot provide detailed individual feedback for every submitted program immediately. Automatic graders partially solve this problem, but they often return a limited message such as “wrong answer” or “test failed”. Such feedback informs the student that the result is incorrect, but it does not always explain the reason or suggest a suitable next learning step. Therefore, programming education needs an approach that connects ICT tools, artificial intelligence and pedagogical feedback.

The purpose of this article is to propose an explainable AI-assisted framework for adaptive feedback in programming courses. The framework is designed for digital learning platforms, laboratory classes and blended programming instruction. It evaluates submitted

code, detects likely misconceptions, generates understandable feedback and recommends the next learning task.

Related Work and Research Gap

Digital learning platforms have made it possible to collect programming submissions, compile logs, test results, time-on-task records and revision histories. These data can be used not only for grading, but also for understanding how students learn programming. Automated programming assessment systems are useful for checking correctness and scalability, while learning analytics can reveal progress patterns across multiple tasks.

Recent artificial intelligence methods, especially transformer-based models, can support code explanation, error localization and natural-language hints. At the same time, direct use of AI-generated feedback without pedagogical control may create new risks: the hint may be too direct, the explanation may be inaccurate, or the student may receive an answer instead of guidance. For this reason, AI support in programming courses should be explainable, controlled and aligned with learning objectives.

The research gap is the lack of a compact instructional model that links code correctness, misconception diagnosis, feedback clarity and adaptive task recommendation in one explainable cycle. The proposed framework addresses this gap by treating feedback as a pedagogical decision, not only as a technical output of an automatic grader.

Proposed Framework

The proposed framework is shown in Figure 1. First, a student submits code together with task information. Second, the system performs pre-processing: tokenization, syntax tree extraction, execution log collection and test-case matching. Third, static analysis and dynamic testing identify syntax mistakes, runtime errors, failed tests, inefficient fragments and style problems. Fourth, the AI diagnosis module estimates the likely misconception behind the error. Fifth, the feedback generator produces a short explanation, a guiding hint and a recommendation for the next task.

The main pedagogical principle of the framework is to avoid giving the final solution immediately. The system first explains the error type, then gives a hint, then suggests a smaller subtask or a similar exercise. The teacher can review the dashboard and adjust the feedback strategy when necessary. Thus, the model supports teacher decision-making and preserves the educational role of human supervision.

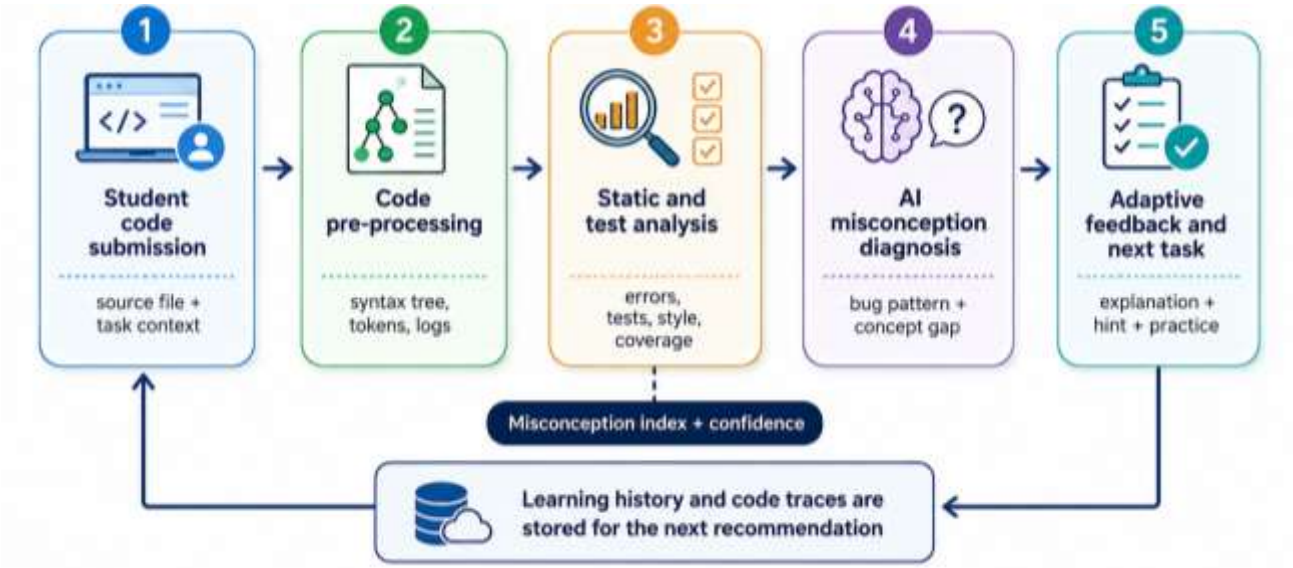

Figure 1. Proposed explainable AI pipeline for adaptive feedback in programming courses.

Mathematical Model

The framework uses five interpretable indicators. S denotes syntax correctness, T denotes test-case success, L denotes logical structure quality, C denotes code-style and readability, and P denotes previous learning progress. The misconception pressure score is calculated as follows:

$$M = \beta_0 + \beta_1(1 - S) + \beta_2(1 - T) + \beta_3(1 - L) + \beta_4(1 - C) - \beta_5 P \quad (1)$$

In Eq. (1), low syntax correctness, failed tests, weak logical structure and poor readability increase the misconception pressure. Previous progress decreases the pressure because the system considers that a strong student may need a shorter hint, while a beginner may need a more detailed explanation. The final feedback urgency is obtained through the sigmoid function:

$$F = 1 / (1 + \exp(-M)) \quad (2)$$

The value of F is interpreted through three levels: low urgency if $0 \leq F < 0.35$, medium urgency if $0.35 \leq F < 0.70$, and high urgency if $0.70 \leq F \leq 1$. The recommendation score for the next task is calculated by combining the current misconception and the difficulty of available exercises:

$$R_k = \gamma_1(1 - |D_k - F|) + \gamma_2 A_k + \gamma_3 G_k \quad (3)$$

Here, D_k is the difficulty level of task k, A_k is its alignment with the current learning goal, and G_k is its usefulness for closing the detected concept gap. The system recommends the task with the highest R_k value.

Figure 2 illustrates an AI-assisted programming feedback dashboard that combines explainable AI workflows with adaptive learning analytics. The system analyzes submitted

Python code, detects errors, identifies concept gaps, estimates confidence, and provides targeted hints, next tasks, and progress indicators to support personalized programming learning.

Figure 2. Example of an AI-assisted programming feedback dashboard.

5. Results and Analytical Discussion

A scenario-based analytical comparison was prepared for four feedback approaches: manual feedback only, a static automatic grader, a general AI chatbot, and the proposed explainable adaptive framework. The comparison considers feedback speed, explanation quality, pedagogical control and support for individualized learning.

Table 1. Analytical comparison of programming feedback approaches.

Feedback approach	Feedback speed	Explanation quality	Pedagogical control	Adaptive recommendation
Manual teacher feedback	Low	High	High	Medium
Static automatic grader	High	Low	Medium	Low
General AI chatbot	High	Medium	Low	Medium

Proposed explainable AI framework	High	High	High	High

The comparison shows that manual feedback is pedagogically rich but time-consuming. A static automatic grader is fast, but it does not fully explain why the solution fails. A general AI chatbot can generate explanations, but without instructional constraints it may provide excessive hints or direct answers. The proposed framework combines automatic analysis with explainability and teacher-oriented control. This makes it suitable for laboratory classes where students need fast guidance but teachers still need visibility into the learning process.

The dashboard in Figure 2 illustrates how feedback can be presented in a transparent way. The student sees the detected issue, the confidence level, a short hint and a recommended next task. The teacher can use the same information to identify common misconceptions across the group and prepare additional exercises.

Scientific Novelty and Practical Value

The scientific novelty of the study is that feedback in programming education is formulated as an explainable AI-supported pedagogical decision rather than a simple grading procedure. The framework integrates code analysis, test results, learning progress and misconception diagnosis into one adaptive feedback model.

The second novelty is the introduction of an interpretable misconception pressure score. This score helps determine whether the student needs a short hint, a detailed explanation or a simpler preparatory task. The third novelty is the combination of adaptive task recommendation with teacher supervision.

The model does not aim to replace the teacher; it supports the teacher by providing structured evidence about student difficulties.

The practical value of the framework is that it can be implemented in programming laboratories, online learning platforms, university learning management systems and coding bootcamp environments.

It may reduce delayed feedback, support individual learning paths and improve the quality of formative assessment in programming courses.

Conclusion

This paper proposed an explainable AI-assisted framework for adaptive feedback in programming courses.

The framework combines code pre-processing, static and dynamic analysis, misconception diagnosis, natural-language feedback and adaptive task recommendation.

The proposed model supports both students and teachers: students receive understandable guidance, while teachers obtain analytical information about common errors and learning gaps.

The analytical comparison indicates that the proposed framework can provide a better balance between speed, explanation quality and pedagogical control than manual feedback, static graders or unrestricted AI chatbots alone. Future work should validate the framework with real student submissions, compare its effect on learning outcomes and develop domain-specific feedback rules for different programming languages.

REFERENCES:

1. Nurjabova D., Sayyora Q., Gulmira P. Using Discretization and Numerical Methods of Problem 1D-3D-1D Model for Blood Vessel Walls with Navier-Stokes //International Conference on Next Generation Wired/Wireless Networking. – Cham : Springer Nature Switzerland, 2022. – С. 73-82.

2. ugli Norboev B. U. et al. Artificial Intelligence in Education and the Digital Preconditions of Tertiary Expansion in Uzbekistan: ARDL Evidence from 2000–2023. – 2026.

3. Ochilova S., Berdiyev G., Xujaqulov N. Fraktal nazariyasiga asoslangan musiqa kompozitsiyasining tahliliy usullari //Journal of Transport. – 2025. – Т. 2. – №. 3. – С. 136-139

4. Rashidovich B. G. DESIGNING FRACTAL BUILDINGS USING ITERATIVE FUNCTION SYSTEMS.

5. Alimov U., Zohirov Q., Berdiyev G. WAYS TO DEVELOP COORDINATION SKILLS IN KURASH //Mental Enlightenment Scientific-Methodological Journal. – 2024. – Т. 5. – №. 09. – С. 9-14.