WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Development of a Regular Assessment Software to Support the Teaching of Fundamental Engineering Courses

Pham Van Duyet [1], Nguyen Van Vu [2*]

[1] *Department of Logistics & Techniques, AD-AF Academy of Vietnam, Vietnam*
[2] *Faculty of Fundamental Technical, AD-AF Academy of Vietnam, Vietnam*
[*] *lengocgianglinh@gmail.com*

A B S T R A C T

In the ongoing reform of higher education, the enhancement of regular assessment plays a crucial role in the early identification of students' learning difficulties, enabling timely adjustments to teaching and learning activities to improve training quality. This is particularly essential for fundamental engineering courses, which serve as the foundation for subsequent specialized subjects. Assessments in these courses must not only provide quantitative measures but also support the development of students' technical thinking. However, in traditional learning environments, the organization of regular assessments faces several limitations, such as the lack of tools for managing question banks, time-consuming test administration, and difficulties in compiling results and evaluating learners' competencies according to learning outcomes.

This paper presents the research, design, and development of a regular assessment software system aimed at supporting the teaching of fundamental engineering courses in a modern direction. The software integrates information technology and contributes to digital transformation goals in education. It enables instructors to build a rich question bank organized by course topics and facilitates the administration of individual or group tests in the form of objective multiple-choice questions. The system is capable of automatically evaluating test results and generating detailed reports, allowing instructors to provide timely feedback to learners. Additionally, the software supports knowledge reinforcement through randomized test sets, enabling students to conduct self-review and improve their academic performance.

Pilot implementation of the software in several courses, such as Electrical and Radio Measurement, Data Transmission Techniques at the Department of Measurement, Faculty of Fundamental Technical, AD-AF Academy of Vietnam, has shown promising results: students were more engaged in learning activities, test results were updated in real-time, and instructors could readily identify learning gaps and adjust instructional content accordingly. The software thus contributes to enhancing transparency and fairness in assessment, while improving the overall quality of training. The paper also suggests future development directions, such as integration with Learning Management Systems (LMS) and the development of a smart dashboard for visualizing student learning progress.

Keywords: Regular Assessment Software, Fundamental Engineering Courses, Digital transformation.

1. Introduction

Digital transformation is emerging as a key driving force in promoting innovation in higher education, particularly in assessment practices. For fundamental engineering courses—which serve as a critical foundation in the training programs for technical personnel—the need to innovate assessment methods has become increasingly urgent in order to enhance teaching quality, meet learning outcome standards, and address practical professional demands.

Recent studies have shown that integrating information technology into learning assessment, especially through automated testing systems, not only reduces the workload for instructors but also enables timely and objective feedback for learners [1], [8], [12]. Moreover, the development of Learning Analytics Dashboards (LADs) has introduced new approaches to monitoring student progress, identifying learning issues early, and supporting personalized learning paths [5], [6], [7], [9], [13].

In the field of engineering education, particularly in fundamental engineering courses, automated assessment tools such as GitSEED have demonstrated their effectiveness in tracking student progress, ensuring fair grading, and supporting outcome-based evaluation [1]. At the same time, the integration of learning analytics and instructional design has shown positive impacts on learners' academic performance [14].

However, to fully harness the potential of such tools in practical teaching environments, it is essential to tailor them to the specific characteristics of each course and learner group, while ensuring adherence to ethical standards in assessment and the use of artificial intelligence in education [4]. The

development of regular assessment software that is flexible, user-friendly, capable of integrating learning analytics, and providing timely feedback is becoming a promising direction for research and application [3], [10], [11].

Furthermore, studies on assessing creative competence in engineering [2] have emphasized that assessment tools should aim to foster creative thinking and problem-solving skills, rather than merely testing knowledge retention.

Driven by these practical demands and research directions, this paper presents the development process of a regular assessment software system tailored to fundamental engineering courses, with a focus on automation, interactivity, and adaptability to diverse learner groups. The paper also proposes an application model suited to the current training context in military academies and universities.

2. Theoretical and Technological Foundations

2.1 Assessment Models in Modern Education

In contemporary education, assessment is no longer solely aimed at measuring learners' knowledge; it also plays a vital role in supporting the learning process. Formative assessment models emphasize the provision of continuous feedback, enabling learners to identify their strengths and weaknesses and adjust their learning strategies accordingly. The adoption of automation in assessment reduces instructors' workload, enhances objectivity, and facilitates timely feedback delivery, thereby improving educational effectiveness [3], [8].

Modern automated assessment systems are capable not only of generating and scoring tests automatically but also of integrating learning analytics to support personalized evaluation. This has become an essential trend in the digital transformation of education, particularly suitable for foundational engineering courses that demand precision and efficiency in assessment.

2,2 Rationale for Using Python and the Tkinter Library

Python was selected as the primary programming language for the software development due to its clear syntax, extensive library support, and high scalability. In particular, Python has a strong developer community and provides powerful tools for data processing, user interface design, and analytics integration.

Tkinter, Python's standard library for building graphical user interfaces, was chosen for its lightweight nature, ease of use, and suitability for applications requiring simple and user-friendly interfaces. The use of Tkinter shortens development time and ensures cross-platform compatibility without requiring complex installations.

2.3 Word File Handling via the python-docx Library

To enable flexible management of question banks and test content, the software incorporates functionality for reading, writing, and editing Word (.docx) documents using the python-docx library. This library allows direct manipulation of Word file structures, facilitating the convenient input of questions, answers, and other related content while preserving original formatting.

The use of python-docx supports the automation of test creation, efficient storage of question data, and synchronization between instructional documents and the testing software—ultimately improving the effectiveness of instructional resource management.

2.4 Mechanisms for Randomization, Timing, and Result Display

The software employs a randomization mechanism for both question selection and answer order to ensure fairness and variation across test instances. This randomization minimizes cheating, increases the challenge level, and enhances the validity of performance assessments.

A built-in timing feature tracks and regulates the duration of each test, ensuring discipline and standardizing testing times in accordance with the requirements of specific courses. This function also enables the system to automatically end the test upon expiration of the allocated time, ensuring fairness for all examinees.

Upon test completion, the software automatically compiles and displays the results, including scores, correct and incorrect responses, and optional feedback comments. Immediate result visualization allows learners to promptly understand their performance and supports instructors in evaluating and adjusting their teaching plans accordingly.

3. Software Design and Implementation

3.1 System Architecture

The multiple-choice test software is designed using a modular architecture, consisting of the following main components:

Input Interface: Allows users to specify the number of questions to be included in the test and the time allocated per question (in minutes). Built using the Tkinter library, this interface provides intuitive data entry and validation for user inputs.

DOCX File Reader: Utilizes the python-docx library to read and parse Word documents containing question banks. Each question is extracted based on a predefined format marked with "Câu" (Question) and followed by multiple-choice options (A, B, C, D), with the correct answer recognized and stored.

Randomized Test Generation: From the parsed question set, the software randomly selects the required number of questions, ensuring variation and uniqueness for each test instance.

Question Display and Answer Selection: Questions are displayed within a scrollable frame, allowing users to read and select answers using radio buttons. Each question is presented clearly, supporting multi-line text with automatic line wrapping.

Exam Timer: The total exam time is computed by multiplying the number of questions by the time per question. A countdown timer updates in real-time and is prominently displayed for users to track progress.

Scoring and Result Display: Upon test completion or when time expires, the software automatically grades the test, marks correct and incorrect answers on the interface, and displays the total score and the number of correct answers in a separate popup window.

Retake Functionality: Users may choose to retake the test, in which case a new random set of questions is generated and the timer is reset.

3.2 Processing Workflow

The software's test-processing workflow involves the following key steps:

Software Launch: The application starts with an input interface prompting users to enter the number of questions and the time per question.

Question File Selection: Users select a .docx file containing the question bank. The software reads and parses the file, extracting questions according to the predefined structure.

Input Validation: The software verifies that the requested number of questions does not exceed the available questions in the file, and that the time values are valid (greater than 0).

Random Test Generation: A randomized subset of questions is selected from the parsed list.

Test Display: The selected questions are displayed on the main interface, with answer choices and a global countdown timer.

Answering and Time Monitoring: Users select answers within the allotted time. If time expires, the test is automatically submitted.

Submission and Grading: The software compares the user's answers with the correct ones, marks each question accordingly, and displays the total score and number of correct answers.

Option to Retake: Users may opt to retake the test, in which case the process restarts with a new randomized question set.

This process can be illustrated by the following flowchart:

Start → Enter number of questions and time per question → Select .docx file → Read questions → Validate input → Generate random test → Display test and start timer → User selects answers → Time expires or user submits → Grade test → Display results → Retake? → Yes: regenerate test → No: End.

3.3 Sample Code Snippets

a) Reading Questions from a .docx File

The following code snippet utilizes the python-docx library to extract multiple-choice questions from a Word file:

```
def read_questions_from_docx(file_path):
    doc = Document(file_path)
    paragraphs = [p.text.strip() for p in doc.paragraphs if p.text.strip()]
    questions = []
    i = 0
    while i < len(paragraphs):
        if paragraphs[i].startswith("question"):  # Detect lines starting with "Question"
            question_text = paragraphs[i]
```

```
            options = []
            answer = None
            j = 1
            while i + j < len(paragraphs):
                line = paragraphs[i + j]
                if line[0] in ["A", "B", "C", "D"] and line[1] == '.':
                    options.append(line)
                elif "Đáp án:" in line:  # Detect the correct answer line
                    answer = line.split("answer:")[-1].strip().upper()
                    break
                j += 1
            if len(options) >= 3 and answer:
                questions.append({
                    "question": question_text,
                    "options": options,
                    "answer": answer
                })
                i += j + 1
            else:
                i += 1
        else:
            i += 1
return questions
```

This function reads all paragraphs in the document, identifies questions that begin with the keyword "Câu" (meaning "Question" in Vietnamese), extracts the answer choices and the correct answer, and stores the information in a dictionary format.

b) Generating Randomized Questions

Random selection of questions from the question bank is handled using the random.sample function to ensure non-repeating and varied test sets:

```
self.questions = random.sample(questions, self.num_questions)
```

This guarantees that each test instance presents a different question set, increasing objectivity and testing efficiency.

c) Displaying Results and Processing Answer Selection

After the user submits the test, the software compares each selected answer with the correct one, marks the result directly on the interface, and calculates the total score:

```
def submit(self):
    correct = 0
    for i, q in enumerate(self.questions):
        selected = self.vars[i].get()
        self.user_answers[i] = selected
        if selected == q['answer']:
            correct += 1
```

```
            self.labels_answer_keys[i].config(text="✅ CORRECT", fg="green")
        else:
            self.labels_answer_keys[i].config(
                text=f"❌ INCORRECT - CORRECT ANSWER: {q['answer']}",
                fg="red"
            )
    score_per_question = 10 / self.num_questions
    total_score = round(correct * score_per_question, 2)
    # Display result popup
    ...
```

The system provides accurate, intuitive, and fair evaluation for the learner by visually indicating correct and incorrect answers and computing a precise score.

4. Pilot Implementation Results

The periodic testing software was piloted at the Department of Measurement, AD-AF Academy during the second semester of the 2024–2025 academic year. It was applied to engineering classes specializing in aviation, radar, missiles, and anti-aircraft artillery. The trial was conducted across three classes with a total of over 100 students. Each class used the software to administer periodic assessments for the courses "Electrical and Radio Measurement" and "Data Transmission Technology."

The software was installed directly on classroom computers or used on the instructor's personal computer. Instructors only needed to prepare a test file in .docx format following a specified template and input key parameters such as the number of questions and time per question. The software automatically generated a randomized exam for each student, provided a clear testing interface with a countdown timer, supported answer selection, and delivered instant feedback upon submission.

4.1. Feedback from Instructors

Instructors involved in the pilot highly appreciated the convenience and stability of the software. The entire process—from test creation to administration and grading—was fully automated, significantly reducing manual workload compared to traditional methods. The ability to adjust the number of questions and time per question allowed instructors to design exams that matched students' levels and course requirements.

Additionally, the automatic submission feature when time runs out and the immediate display of results along with the correct answers enhanced transparency, enabling instructors to conduct quick, accurate, and fair assessments.

4.2 Feedback from Students

Students who participated in the pilot showed enthusiasm and engagement during the test. The software interface was intuitive and easy to use, with clearly displayed questions that helped students focus on their tasks. Many students appreciated the instant feedback feature, which allowed them to self-assess their performance and identify areas needing further study.

Students also expressed satisfaction with the fairness of the system, as each person received a different set of randomized questions and exam time was strictly monitored.

4.3 Comparison with Traditional Testing Methods

The implementation of the software demonstrated several key advantages over conventional paper-based testing:

Time-saving: The entire process—from test preparation, printing, distribution, grading, to result compilation—was replaced by digital operations, reducing the total testing time from several hours to just a few minutes.

Enhanced objectivity: Randomized question generation and automatic grading eliminated subjective bias, ensuring fairness for all students.

Improved self-directed learning: With immediate score and answer feedback, students could evaluate their understanding and adjust their study strategies effectively. Moreover, the "Retry" function enabled repeated practice, thereby reinforcing their knowledge.

Overall, the pilot results show that the software meets current teaching and assessment needs, supports digital transformation in military education, and contributes to improving training quality at the Air Defense and Air Force Academy.

5. Conclusion and Development Directions

The periodic assessment software developed to support the teaching of foundational engineering courses was designed with clear and specific objectives: to assist instructors in organizing effective, objective assessments of students' learning progress in line with current trends in digital transformation in education. Through its pilot implementation at the Department of Measurement, AD-QF Academy, the software has demonstrated practical effectiveness in various aspects.

First, the software automates the entire assessment process—from test creation, administration, to grading and result feedback—significantly reducing the time and effort required from instructors while ensuring accuracy and transparency.

Second, it enhances objectivity and fairness in evaluation through randomized test generation and automatic scoring, minimizing the subjective influence typically encountered in traditional testing methods.

Third, the software promotes student autonomy by providing immediate feedback, allowing learners to gauge their understanding and encouraging them to review and reinforce their knowledge through the "Retry" feature.

Despite these positive outcomes, the development team recognizes that the software has significant potential for expansion and improvement to better meet the practical needs of teaching and learning. Some proposed development directions include:

Expanding topic-based question banks: Allow for the creation of rich question sets categorized by content area or cognitive level, enabling instructors to flexibly select and combine questions during test creation, while also supporting students in focused practice on specific topics.

User account management with role-based access: Implement separate login systems for instructors and students. Instructors can create and manage tests, and view aggregate results; students can take tests and review their personal performance history. This clear role separation enhances both security and professionalism in software operation.

LAN/Internet connectivity: Expand the software's usability beyond local setups to broader networks, enabling remote testing or simultaneous assessments across multiple classes. This feature is particularly useful in the context of blended and online learning, which is becoming increasingly prevalent.

Integration of result storage, analytics, and score reporting: Establish a database to store the entire test process, including questions, answers, and scores. Based on this data, the software can generate statistical analyses of performance by class, subject, or individual student—supporting instructors in conducting comprehensive learning evaluations. Additionally, features like printable score reports and exportable summaries will meet educational management and reporting needs.

In conclusion, the software has proven its feasibility and effectiveness in administering periodic assessments for foundational engineering courses. Continued development and expansion along the proposed directions will further drive digital transformation in military education, enhance training quality, and better meet the practical demands of educational institutions and the Ministry of National Defense in the current era.

References

[1]. P. Orvalho, M. Janota, and V. Manquinho, "GitSEED: A Git-backed Automated Assessment Tool for Software Engineering and Programming Education," arXiv preprint arXiv:2409.07362, Sep. 2024.

[2]. Z. G. Akdemir-Beveridge, A. Zaghi, and C. Syharat, "Understanding and Evaluating Engineering Creativity: Development and Validation of the Engineering Creativity Assessment Tool (ECAT)," arXiv preprint arXiv:2504.12481, Apr. 2025.

[3]. S. Brdnik, B. Šumak, and V. Podgorelec, "Aligning Learners' Expectations and Performance by Learning Analytics System with a Predictive Model," arXiv preprint arXiv:2211.07729, Nov. 2022.

[4]. M. Perkins, L. Furze, J. Roe, and J. MacVaugh, "The AI Assessment Scale (AIAS): A Framework for Ethical Integration of Generative AI in Educational Assessment," arXiv preprint arXiv:2312.07086, Dec. 2023.

[5]. G. Ramaswami, T. Susnjak, and A. Mathrani, "Effectiveness of a Learning Analytics Dashboard for Increasing Student Engagement Levels," Journal of Learning Analytics, vol. 10, no. 1, pp. 1–20, 2023.

[6]. L. Chen et al., "How Students Use Learning Analytics Dashboards in Higher Education: A Learning Performance Perspective," SAGE Open, vol. 13, no. 3, pp. 1–15, 2023.

[7]. F. Vogel et al., "Team Interactions with Learning Analytics Dashboards," Computers & Education, vol. 182, p. 104514, 2022.

[8]. Q. Hao et al., "Towards Understanding the Effective Design of Automated Formative Feedback for Programming Assignments," Computer Science Education, vol. 32, no. 2, pp. 105–127, 2022.

[9]. P. Leitner, M. Ebner, and M. Ebner, "Learning Analytics Dashboards for Assessing Remote Labs Users' Work: A Case Study with VISIR-DB," Technology, Knowledge and Learning, vol. 29, pp. 1–20, 2024.

[10]. D. Ifenthaler, C. Schumacher, and J. Kuzilek, "Investigating Students' Use of Self-Assessments in Higher Education Using Learning Analytics," Journal of Computer Assisted Learning, vol. 39, no. 1, pp. 255–268, 2023.

[11]. T. Khelifi et al., "EX-LAD: Explainable Learning Analytics Dashboard in Higher Education," presented at the International Conference on Learning Analytics and Knowledge, Jan. 2024.

[12]. Elmahdi, A. Al-Hattami, and H. Fawzi, "Towards a Framework to Support the Implementation of Digital Formative Assessment in Higher Education," Education Sciences, vol. 12, no. 11, p. 823, 2022.

[13]. H. He et al., "LearningViz: A Dashboard for Visualizing, Analyzing and Closing Learning Performance Gaps—A Case Study Approach," Smart Learning Environments, vol. 11, no. 1, pp. 1–20, 2024.

[14]. M. Blumenstein, "Synergies of Learning Analytics and Learning Design: A Systematic Review of Student Outcomes," Journal of Learning Analytics, vol. 7, no. 3, pp. 13–28, 2020