

Design and Implementation of a Closed-Loop Management System for Multi-Source Learning Evidence in Creative Programming Education

Yingjiang Wang

College of Information Engineering, Suqian University, Suqian, Jiangsu, China

Abstract: Creative programming courses generally cover classroom teaching, development of programming artifacts, regular quizzes, teacher grading and learning process documentation. If this data is stored in separate course, artifact and examination modules, it will be difficult for teachers to track the progress of student learning systematically. Dispersal of the evidence also limits the scope of process-oriented evaluation. Therefore, this paper proposes a design and implementation of a closed-loop management system for multi-source learning evidence in creative programming education. The basic organisational unit of instructional activities in the system is a course unit. Link course resources, attendance check-ins, class assignments, programming artifacts, knowledge-point questions, examination answers and error records. The linked data supports different access and management rights for administrators, teachers, students and parents. The management interface uses Vue 3 and Ant Design Vue. The backend services are based on Spring Boot, MyBatis-Plus, MySQL, Redis, Shiro/JWT, and a separate WeChat mini-program service extends query access for parents. For objective questions, the system will automatically grade them and record any incorrect answers upon submission of the exam. For questions of Scratch programming, the external program validation interface will return a similarity score or test result; however, creative tasks and subjective questions will still be graded by the teacher. The system can coordinate the management of learning process records, artifact assessment and knowledge-point diagnosis for creative programming courses. Evaluation is limited to the function and data traceability of the deployed system, and no causal improvement in learning achievement has been established.

Keywords: Creative Programming Education; Process-Oriented Assessment; Learning Evidence; Scratch; Online Teaching Management System

1. Introduction

1.1 Background

Creative programming courses primarily use Scratch, Python, and C++ tasks to teach sequencing, events, conditionals, loops, variables, and algorithmic thinking through project creation [1–3]. Unlike courses centered on lectures and a single written examination, creative programming courses produce learning outcomes beyond quiz scores. These outcomes include classroom participation, course unit completion, artifact files, teacher feedback, and subsequent practice targeting knowledge points associated with errors. Course management systems therefore need to store heterogeneous process data and link them to specific students, course units, and learning tasks.

Many teaching platforms already support course publishing, resource management, or online quizzes. In practice, however, course resources, assignment artifacts, and examination records are often presented separately. Teachers must consult multiple pages to determine whether students attended class, completed their artifacts, received grading, and mastered the relevant knowledge points. This arrangement increases the operational burden of process-oriented assessment and makes it difficult to maintain a continuous feedback loop. Studies of learning situation analysis and student status management systems identify multi-source educational data organization, role-specific access, and lightweight frontend interaction as important directions [4–7]. Nevertheless, integrating artifact evidence, program validation, and knowledge-point-based quizzes for creative programming courses remains a practical engineering challenge. This study examines a deployed creative

programming platform and designs a closed-loop mechanism for managing multi-source learning evidence. Workflow links course units, learning activities, artifact submissions, quiz-based diagnoses, error accumulation and teacher feedback. It does not automatically predict or decide on learning for students; instead, it addresses the problem of delayed feedback and subsequent adjustments by learners in formative assessment [8-10]. The system will have an organised record of the teaching process and assessment support for teachers to review.

1.2 Scope and Contributions

The main results of this study are as follows:

- (1) A closed-loop model of multi-source learning evidence for creative programming courses is constructed. Course units serve as the central hubs for all other resources, attendance records, assignments, artifacts, quizzes, response logs and error logs in a unified teaching process.
- (2) A management system is used to organise courses, evaluate artifacts, conduct knowledge-point-based quizzes and record learning progress. Support the submission and teacher grading of Scratch, Python and C++ artifacts; automatic scoring for objective questions; manual grading for subjective questions; and external validation of selected programming questions.
- (3) Functional tests will be conducted on a system deployed in the public Internet. Both the artifact management and the grade management modules have more than 1,000 records; therefore, the evidence of the learning process in the authentic teaching environment is well-preserved and well-organised.

2. Requirements Analysis and Overall Design

2.1 User Roles and Functional Requirements

The three principal users are administrators, teachers and students. Administrators can centrally manage users, roles, classes, courses, question banks and system settings. Teachers post course units and homework, collect students' homework, grade the homework, and view test papers. Students can view their courses, log in at the appropriate time, submit programming artifacts, take quizzes, and view their learning history. The parent interface provides lightweight functions for parents linked to student accounts, including lesson-hour summaries, lesson-hour adjustment records, and artifact queries. A course unit is the smallest manageable

instructional activity in the system. Each unit can include videos, examples, courseware, assignments, start and end times, an attendance check-in option, and an in-class practice examination. Students check in and submit artifacts within the unit; teachers assign scores, provide feedback, or add tags to the artifacts. When a quiz is configured, student responses also generate counts of correct and incorrect answers, scores, and error information. The system therefore requires cross-module associations based on identity, course membership, and time.

2.2 Closed-Loop Model of Multi-Source Learning Evidence

The system has four kinds of learning materials. Evidence of instructional provision includes courses, course units, videos, courseware, examples and task descriptions. Participation evidence includes the time and method of check-in, and the status of lesson-hour adjustments. Artifact-based practice evidence includes uploaded artifact files, cover images, submission sources and statuses, views and likes. Knowledge diagnosis evidence includes the knowledge points related to the questions, submitted answers, objective-question scores, teacher feedback, counts of correct and incorrect answers, and cumulative error counts.

The closed loop is designed to maintain traceable relationships among instructional tasks. After resources and tasks are published within a course unit, students generate participation and practice records through check-ins and artifact submissions. Teacher grading produces assessment records, while participation in linked examinations generates response records. Incorrectly answered questions are added to each student's personal error log. Teachers can then use this evidence to adjust courses, assignments, or review schedules. This design makes artifact assessment and knowledge-point diagnosis accessible within the same course context.

2.3 Nonfunctional Requirements

The system must provide role-based access isolation, data consistency, file accessibility, and extensibility. Students may access and modify only their own data; teachers access student records within their assigned courses or classes; administrators have global configuration privileges. Examination submission, objective-question scoring, error collection, and record status updates must occur within a single

transaction. The system also requires a consistent API response format, exception handling, and file storage to support access to artifact files, cover images, and learning resources.

3. System Architecture and Data Model

3.1 Overall Architecture

The two parts of the system are the front end and the back end. Vue 3 and Ant Design Vue are used to build the management interface, and it has modules for courses, assignments, examinations, classes, and system settings. Spring Boot is used for the backend to separate the controller, service, DAO and entity layers of the business logic. MyBatis-Plus provides entity persistence and paginated queries; MySQL is used for the main business data; Redis supports caching and authentication. Shiro and JSON Web Tokens (JWT) are used for login authentication and role-based access control (RBAC); Swagger provides API documentation. The front-end and back-end

communicate to obtain resources via RESTful APIs. Business modules are separated by responsibility to reduce API coupling and support future service-oriented extensions [11-13].

For mobile access, a separate WeChat mini-program and a Node.js/Express service are provided by the system. This service verifies parent identities and retrieves lesson-hour summaries, lesson-hour adjustment records, and artifacts according to parent–student associations. The sets of exposed data and interface complexities for the management, student and parent interfaces are different; however, they all use the same core course, student and artifact data. Figure 1 is the general system architecture. The management, student and parent interfaces access the Spring Boot business services via HTTPS/REST APIs. These services connect to MySQL, Redis and file storage. The programming-question validation module calls an external program validation service via a controlled interface.

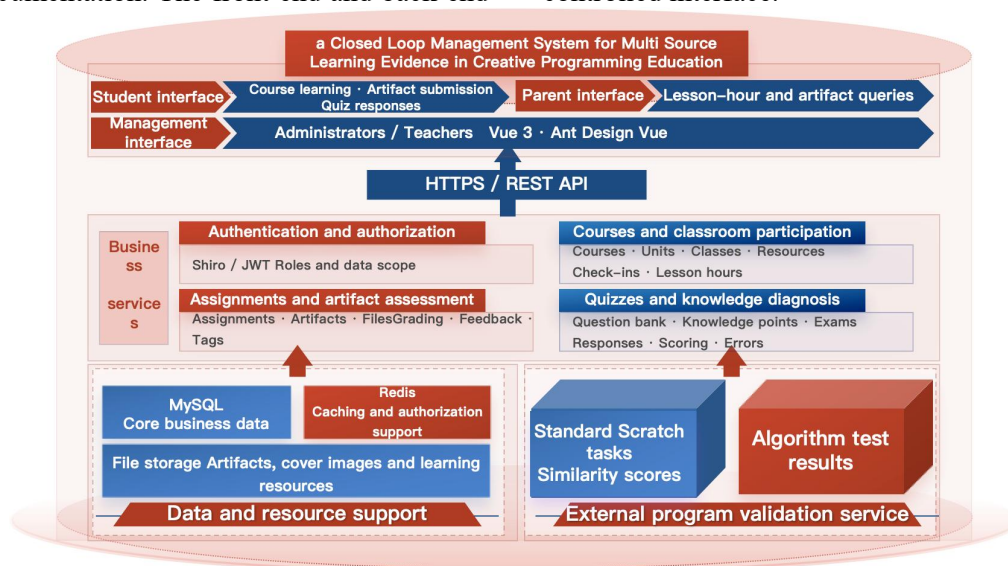


Figure 1. Overall System Architecture

3.2 Core Data Relationships

Course and course unit entities provide the basic structure for organising teaching materials. Each course unit has a name, description, resources, start and end time, attendance check-in settings, and linked examinations. Associations with classes and students define the teaching content. Attendance records link users, courses and course units, and store check-in time, method, IP address and status. The system can link check-in status to lesson-hour adjustment records to obtain proof of attendance.

The artifact entity stores the user, class, course unit, supplementary assignment, artifact name,

type, status, file, cover image, and creation time. Artifacts can be linked to grading and comment records; grading records contain teacher scores and feedback. Examination data are organized around questions, test papers, examinations, examination records, and response records. Each question can be associated with one or more knowledge points. Supported types include single-choice, multiple-choice, true/false, fill-in-the-blank, short-answer, and programming questions. Examination records store total, objective-question, and subjective-question scores; correct and incorrect answer counts; start and submission times; and examination status.

Response and error records retain information at the individual-question level.

3.3 Security and Privacy Design

Data access is restricted through three layers: identity authentication, role-based authorization, and course-scope validation. Following login, Shiro/JWT identifies the user, and the current user identity is propagated through API calls. Menu access and business operation permissions are assigned according to roles, including administrator, teacher, student, and parent. Data-scope controls further restrict access to course, class, and student records. The examination module filters accessible records by user role and organizational department scope. Only administrators, users with the developer role, or course creators can modify the corresponding attendance check-in statuses. Before querying lesson hours or artifacts, the parent interface additionally verifies the parent – student association.

Learning process data include sensitive information such as student identities, course participation, artifact files, and examination scores. Therefore, the system will only use the minimum required amount of data. Students can only view their own records; teachers can only view student data in their teaching scope; parents can only view information of linked students. Previewing and accessing files of artifacts require identity verification and resource permission checks. The design meets the requirements of learning analytics for transparency, purpose limitation, access control and learner privacy [14-15]. It supports the traceability of instructional processes and establishes privacy protection for future learning analytics.

4. Implementation of Core Functional Modules

4.1 Course Unit and Classroom Participation Management

Teachers can set up an organised sequence of course units and arrange videos, examples, courseware, homework, and class time for each unit. Course unit pages show the start and end times and cover images, and organise creative programming content according to project themes. The attendance module will record the time of check-in for a course unit and the time of check-out. Only one attendance record is saved for that user at that time. The available statuses are

awaiting check-in, checked-in, absent, and late. The management interface has attendance queries by class, user, course, course unit and status, and shows the check-in method, time and status. This module connects classroom participation with course units and lesson-hour information to reduce the need for offline roll calls.

4.2 Assignment and Programming Artifact Assessment Management

The modules are: Assignments, Extra Assignments, Independent Creations and Examination Papers. When a student submits an artifact, the system will save its name, type, source, file, cover image, submission time and status. Teachers can preview, view, grade, comment on and tag artifacts in the management interface. Grading summary queries report the expected, received, graded and ungraded submission numbers for a course unit or assignment. These summaries help the teacher grade the students.

Search conditions for the artifact management module are assignment name, account, student name, tag, type and creation source. The list distinguishes between sources, such as independent creations, class assignments, and examination artifacts, and types, such as Scratch, Python, and C++ examination artifacts. Therefore, the system will save and distinguish artifact processing data for different teaching environments.

4.3 Examinations and Automatic Scoring with Error Collection

The management module is for the question bank, knowledge points, test papers, examinations and related records. When a student begins an exam, the system will create an exam record and save the student's answers during the test. The back-end saves the submitted answers based on the type of question. Objective questions are automatically scored based on the reference answers, and teachers must grade the short-answer questions. The system will save the draft of the program. Select the standard Scratch questions and algorithmic questions for external program validation service, and teachers will grade the creative programming questions. Thus, the evaluation of open-ended works will not be reduced to a single, automatically generated score. If the objective question is answered incorrectly or left blank, the system will automatically add it to the list of errors for that student. Increment the

cumulative error counter and record the time of the latest error. Link the error record to the original question information, and then display the question, its knowledge points, and the incorrect answer for teachers and students. The grade management page shows the examination name, examinee, score, count of correct and incorrect answers, grade status, pass/fail status, and submission time. The above are direct indicators of knowledge-point revision and teaching feedback.

4.4 Role-Specific Evidence Access and Feedback

Role-Specific Evidence Access and Feedback Students can view their learning process records in My Courses, My Check-ins, My Assignments and the examination page. Teachers and administrators can view class- or course-level evidence in the course, assignment and grade management sections. The parent interface shows lesson-hour information, artifacts and lesson-hour adjustment records for linked students. The system has a reward-point policy. Submission of the test will be assigned a score and reward points based on learning activities.

The current system can collect and retrieve the process data for assessment, but it does not automatically build a learner profile or offer recommendations. Teachers should still use professional judgment based on the quality of artifacts, classroom performance and errors in specific knowledge areas. Division of labour is suitable for creative programming tasks with open-ended results and evaluation standards that change according to circumstances.

5. System Verification and Application Analysis

5.1 Verification Setting and Procedure

The system is running in the public Internet. Verification includes interface inspection and checks of business workflows. First, course unit pages are inspected to confirm course-based organization and the display of instructional times and resources. Secondly, there is a search function for the artifact list, source differentiation and graded access. Third, check the examination records for scores, the number of correct and incorrect answers, grades, and pass/fail results. Finally, attendance pages are checked for the display of course units, check-in times, methods, and statuses.

Verification employs the deployed system to achieve two objectives. First, verify that the main workflow functions correctly according to the design. The workflows are course organisation, artifact submission and grading; examination scoring and error collection; classroom attendance check-in. The second is to verify that the system can save and retrieve the relevant learning process data after extended use.

5.2 Analysis and Scope of the Findings

Based on the verification results, the deployed system can continuously save and display artifact, quiz, and attendance data. It can also be used to organise courses, grade artifacts and review results in one place. Both the records of artifacts and examination records exceed 1,000, so the system has been applied to multiple demonstration cases.

Verification includes testing the function, completeness of workflow and process-data traceability. As shown in the results, the system can store course, artifact, quiz and attendance data from actual teaching situations and provide teachers with a single query and processing interface. The results are limited to system implementation and practical feasibility. The next evaluation will cover the whole teaching cycle, select a suitable sample, and provide the relevant ethical guarantees. The above evaluations will determine whether the participation and quality of the artifacts meet the goals. Further standardisation of knowledge-point associations can also improve support for complex diagnostic queries and future learning analytics.

6. Conclusion

This paper studies fragmented resource, classroom participation, programming artifact and knowledge assessment data in creative programming courses. Design and implement a closed-loop management system for multi-source learning evidence. Course units are the workflow centres of the system that connect attendance check-ins, assignments, artifact grading, knowledge-point quizzes, response records and personal error logs. Management, student and parent interfaces have different data permissions. Process evidence in the system includes the source of artifacts, grades, scores, counts of correct and incorrect answers, and attendance status.

The contribution is a traceable closed-loop workflow for creative programming teaching. In

the future, standardised knowledge graph associations, artifact assessment rubrics, explainable learning analytics, teacher-reviewed personalised revision recommendations and student privacy protection will be available. Controlled studies would be needed to evaluate educational effects. Learning gains should not be derived solely from platform logs. Add process data and an explicit assessment plan to the evaluation, design a sample, and include research ethics provisions.

References

- [1] RESNICK M, MALONEY J, MONROY-HERNÁNDEZ A, et al. Scratch: Programming for all. *Communications of the ACM*, 2009, 52(11): 60-67.
- [2] GROVER S, PEA R. Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, 2013, 42(1): 38-43.
- [3] SÁEZ-LÓPEZ J M, ROMÁN-GONZÁLEZ M, VÁZQUEZ-CANO E. Visual programming languages integrated across the curriculum in elementary school: A two year case study using “Scratch” in five schools. *Computers & Education*, 2016, 97: 129-141.
- [4] ZHU Y W, XIAO Y, KUANG Y X, et al. Design and implementation of learning situation analysis management system based on WeChat mini program. *Journal of Technology Innovation and Engineering*, 2026, 2(5).
- [5] SUN L, HU X. Design and implementation of an intelligent academic status management system for universities based on augmented reality and large language models. *Journal of Technology Innovation and Engineering*, 2026, 2(6).
- [6] FERGUSON R. Learning analytics: Drivers, developments and challenges. *International Journal of Technology Enhanced Learning*, 2012, 4(5/6): 304-317.
- [7] VIBERG O, HATAKKA M, BÄLTER O, et al. The current landscape of learning analytics in higher education. *Computers in Human Behavior*, 2018, 89: 98-110.
- [8] BLACK P, WILIAM D. Assessment and classroom learning. *Assessment in Education: Principles, Policy & Practice*, 1998, 5(1): 7-74.
- [9] SHUTE V J. Focus on formative feedback. *Review of Educational Research*, 2008, 78(1): 153-189.
- [10] NICOL D J, MACFARLANE-DICK D. Formative assessment and self-regulated learning: A model and seven principles of good feedback practice. *Studies in Higher Education*, 2006, 31(2): 199-218.
- [11] MENG Q. Design and implementation analysis of enterprise employee personnel information management system. *Journal of Technology Innovation and Engineering*, 2026, 2(1).
- [12] PAUTASSO C, ZIMMERMANN O, LEYMANN F. RESTful web services vs. “big” web services: Making the right architectural decision//*Proceedings of the 17th International Conference on World Wide Web*. New York: Association for Computing Machinery, 2008: 805-814.
- [13] DRAGONI N, Giallorenzo S, LAFUENTE A L, et al. Microservices: Yesterday, today, and tomorrow//*Present and Ulterior Software Engineering*. Cham: Springer, 2017: 195-216.
- [14] PARDO A, SIEMENS G. Ethical and privacy principles for learning analytics. *British Journal of Educational Technology*, 2014, 45(3): 438-450.
- [15] DRACHSLER H, GRELLER W. Privacy and analytics: It ’ s a DELICATE issue//*Proceedings of the Sixth International Conference on Learning Analytics & Knowledge*. New York: Association for Computing Machinery, 2016: 89-98.