European Journal of Education Science

Vol. 1 No. 3 2025

Article Open Access

Design and Practice of AI Intelligent Mentor System for DevOps Education

Zhengrui Lu [1,*]

[1] Oracle, Seattle, WA, 98101, USA
[*] Correspondence: Zhengrui Lu, Oracle, Seattle, WA, 98101, USA

Received: 12 October 2025
Revised: 29 October 2025
Accepted: 21 November 2025
Published: 29 November 2025

Abstract: The design and application of the AI intelligent mentor system are aimed at meeting the support requirements for immediate code writing, build testing, and deployment and operation processes in DevOps education. Among them, the mixed characteristics of multi-source data and the correlation characteristics of the operation process make it difficult for traditional teaching to meet the requirements of refined immediate response. Therefore, this article explores how to apply AI to support an intelligent mentor system that can meet the requirements. First, describe the characteristics of code base records, pipeline logs, and operation monitoring data, summarize the technical support for system construction and application, and propose specific plans for the overall system architecture, functional module construction, model and algorithm design, etc. At the same time, provide mathematical representations Then discuss how the system responds to issues such as the execution status of the task chain, feedback on the operation process, and learning management during the specific execution training process, thereby providing a useful reference for constructing an intelligent support system with the characteristics of DevOps education.

Keywords: DevOps education; AI intelligent mentor system; system design; technical practice

1. Introduction

DevOps education centers on engineering practices. The learning process encompasses a series of continuous task flows such as coding, building and testing, release and maintenance. The demand for real-time guidance and accurate feedback is becoming increasingly evident. However, when learning behavior data possesses diversity and process-oriented attributes, traditional support methods are difficult to meet the requirements of cognitive tracking and automated analysis in complex technical processes. The AI intelligent coaching system can provide a sustainable technical path for intervention in DevOps scenarios through natural language processing, data analysis, and automatic responses. This article focuses on the educational characteristics and technical requirements of DevOps, constructs a systematic architectural solution, and simultaneously attempts to apply this system in the actual workflow.

1.1. Research Foundation of AI Intelligent Mentor System for DevOps Education

1.1.1. Analysis of Data Structures and Task Characteristics in DevOps Learning Scenarios

DevOps learning tasks have a highly "process-oriented" feature. Each code submission, build execution, pipeline trigger, or container start generates a behavior record with a timestamp, forming a traceable task chain (As shown in Table 1).

Table 1. Typical Data Types in DevOps Learning Scenario.

Data type	Main source	Data characteristics	Supportedteachingpurposes
Code text	Git repository	Semantic structure and style features	Code quality analysis, semantic diagnosis
Build log CI tools (such as Jenkins)	Build log CI tools (such as Jenkins)	Timing, error stack	Build failure location and error explanation
Test results	Automated testing framework	Pass rate, coverage rate	Behavioral inference, ability diagnosis
Deploy data	Docker/K8s	Status code, delay	Deploy anomaly identification and prompt generation

The above-mentioned data are not generated independently of each other, but rather form a chain gradually established as the task progresses. To reflect the logical relationship of data flow from source code to the deployment process, a devops task chain structure diagram was drawn, as shown in Figure 1.

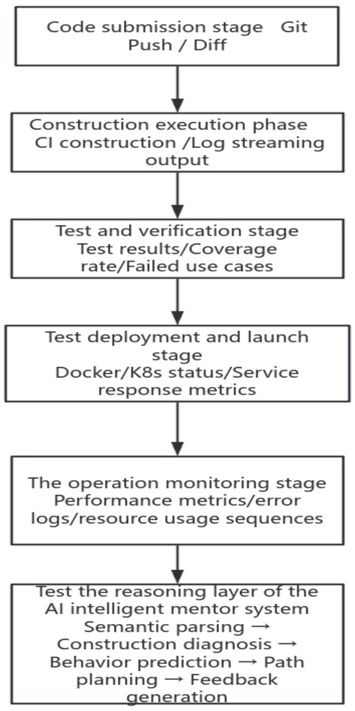
Code submission stage Git Push / Diff
Construction execution phase CI construction /Log streaming output
Test and verification stage Test results/Coverage rate/Failed use cases
Test deployment and launch stage Docker/K8s status/Service response metrics
The operation monitoring stage Performance metrics/error logs/resource usage sequences
Test the reasoning layer of the AI intelligent mentor system Semantic parsing → Construction diagnosis → Behavior prediction → Path planning → Feedback generation

Figure 1. illustrates the data flow structure of the DevOps learning task chain.

Each operation of the learner forms a forward chain of events, and the results of each event can also provide feedback on the previous ones, constituting an iterative learning process. For the intelligent mentor system, this chained data structure enables it to make inferences based on the complete context, thereby supporting fine-grained analysis of the learning process.

1.1.2. Core Technical Foundation and Application Conditions of AI Intelligent Mentor System

The operation of the AI intelligent mentor system in DevOps education relies on multiple underlying technologies. For instance, language processing can interpret code text, error stacks, and log information to help the system identify build and deployment issues. Knowledge retrieval and language generation technologies support the system in generating interpretive feedback, such as error causes and providing modification suggestions, to enhance the operability of the feedback [1]. To ensure the stable operation of the system in a complex DevOps environment, the underlying environment must provide real-time readable data streams to support continuous updates of build logs and operation monitoring. The platform must have stable integration capabilities with toolchains such as Git, Jenkins, and Kubernetes. Model reasoning needs to maintain interpretability, ensuring that feedback is consistent with the operational context. At the same time, the system also needs to adapt to the immediate guidance requirements in multi-person collaborative scenarios under low latency conditions under heavy loads.

2. Design of AI Intelligent Mentor System for DevOps Education

2.1. The Overall Framework and Hierarchical Structure Construction of the System

The overall system architecture adopts a hierarchical design. Through the technical chain of "data access - data processing - model reasoning - interactive presentation", it serializes and structures data from different sources in the DevOps environment and transmits it to the reasoning stage for intelligent analysis. From the perspective of system architecture, the system acquires encoded information, builds log data, detects results and deploys monitoring data in the form of tool interfaces. After the abstraction processing of the received data by the parsing module, vectors or construction features acceptable to the model are formed. Based on their input, the model inference stage conducts capability assessment, fault diagnosis, and work plan formulation. Finally, they are presented to the learner in the presentation layer in the form of structured text, visual cards, or conversational output. The overall structure of the system can be abstracted as a four-layer set, and its formal definition is as follows:

$$\text{System} = \{L_{\text{in}}, L_{\text{process}}, L_{\text{model}}, L_{\text{view}}\} \quad (1)$$

Among them, System is the abstract collection structure of the entire AI intelligent mentor system. L_{in} is the data access layer; $L_{process}$ is the data processing layer; L_{model} is the model inference layer; L_{view} is the interaction presentation layer. Messages are exchanged between layers through asynchronous events to avoid inference delays or data blocking caused by structural coupling. In actual design, the system also needs to support the service-oriented deployment of models, enabling different models to be independently expanded, updated, and called through a unified interface during inference. This service-oriented approach ensures that the architecture has good module division capabilities and horizontal scalability [2].

2.2. Systematic Construction of Core Functional Modules

The core module system is built along the DevOps task chain, including functional modules such as task parsing, behavior diagnosis, feedback generation, and path planning. They form a continuous execution chain through the connection of data and events. The parsing process is modeled in formal expression as:

$$F_{\text{task}} = \text{Parse}(D_{\text{raw}}, \text{Context}) \quad (2)$$

Among them, F_{task} is the structured result after task parsing; Parse (·) for task parsing operations; Draw represents the original data; Context represents the current state of the task chain. The behavior diagnosis module is responsible for identifying abnormal behaviors of learners in DevOps operations, such as build failures and deployment performance degradation. This module usually determines the state in combination with serialized feature vectors, and its calculation method can be expressed as:

$$Diag = \sigma(W \cdot h_{event} + b) \quad (3)$$

Among them, $Diagh_{event}$ is the event embedding vector; σ is the activation function; W is the weight matrix; h_{event} is the event feature vector; b is the bias term, which is used to adjust the activation threshold of the model. The feedback generation module generates explanatory content from the knowledge base or model inference based on the diagnostic results. Its essence is to map abnormal signs into structured output. The feedback generation mechanism can be expressed as:

$$\text{Feedback} = \text{Gen}(\text{Diag}, \text{KB}) \quad (4)$$

Among them, Feedback is the feedback content of the output; Gen (·) is the feedback generating function; $Diag$ is the output of the behavior diagnosis module; KB stands for Knowledge Base. The path planning module is used to determine the optimal action that learners should perform in the next stage of the task chain, and its operation depends on the matching relationship between the ability vector and the task difficulty vector.

2.3. The Composition Mechanism of AI Models and Key Algorithms

The core capabilities of the AI intelligent mentor system are concentrated at the model level, including models such as capability representation, modeling diagnosis, sequence prediction, and task recommendation. In light of the characteristics of the DevOps process, the system adopts mechanisms such as vectorized capability representation, semantic classification, sequence prediction, and similarity matching to construct a complete reasoning framework. The learner's ability is composed of five dimensions, covering coding, building, testing, deployment and monitoring capabilities, and is uniformly represented in the form of a five-dimensional vector:

$$C = [c_{\text{code}}, c_{\text{build}}, c_{\text{test}}, c_{\text{deploy}}, c_{\text{monitor}}] \quad (5)$$

Among them, C is the comprehensive ability vector of the learner; c_{code} is the capability value for code writing and code quality. c_{build} is the ability value for building tasks; c_{test} represents the test capability value. c_{deploy} represents the deployment capability value; $c_{monitor}$ represents the capability value for operation monitoring. This vector is dynamically updated based on the result after each task execution.

In terms of construction diagnosis, the construction logs, after being embedded, can be used as input for the classification model to predict whether the construction is normal or belongs to a certain type of failure. Its classification calculation can be expressed as:

$$y = \text{Softmax}(W_1 \cdot h_{\log} + b_1) \quad (6)$$

Among them, y represents the prediction result of the construction state; $Softmax$ ( ) is the activation function that normalizes the linear output of the model to a probability distribution; W_1 is the weight matrix of the log classification model; h_{log} is a log embedding vector; b_1 is the bias term of the log classification model. The task recommendation model generates recommendation values based on the similarity between the ability vector and the task requirement vector, combined with the task difficulty, thereby achieving personalized task planning. Its scoring formula is as follows:

$$\text{Score}(\text{task}_i) = \alpha \cdot \text{Sim}(C, R_i) + \beta \cdot \text{Diff}(\text{task}_i) \quad (7)$$

Among them, Score (task_i) is the final recommended score of the learning task $task_i$; α and β are the weight coefficients; $Sim\,(C,R_i)$ represents the similarity between the learner's ability vector C and the task requirement vector R_i. R_i is the requirement vector of $task_i$; $Diff\,(task_i)$ represents the task difficulty item.

2.4. Organization Mode of System Data Flow and Interaction Logic

The system has constructed a closed loop of "event - data pipeline - reasoning and decision-making - presentation and feedback". Unlike some static information assistance, this design attaches great importance to the real-time feedback of the system. Therefore, the feedback needs to be promptly manifested as students perform skill actions. Take the learner's action and behavior events as the starting point of learning [3]. By capturing and detecting the events of learners and converting them into processable information fragments, and then processing them layer by layer in the data pipeline to become the features required by the model. The analyzed data is passed through pipelines to the model inference module, and inference results are derived based on classification, sequence prediction, and correlation comparison, etc. The inference results are then organized by the presentation layer into explanatory feedback, thus forming a complete closed loop. After each round of feedback is generated, the system updates the learner's state vector, enabling the next round of reasoning to make judgments based on the latest context. As operations accumulate continuously, the system's understanding of learners' behaviors gradually strengthens, eventually forming a sustainable and iterative intelligent support structure.

3. The practical Application of the AI Intelligent Mentor System in DevOps Education

3.1. Application Methods of the System in the DevOps Training Process

In the DevOps training process, every operation of the learner generates an event with a timestamp. The system continuously listens to these events to achieve real-time intervention in the process. When code changes occur, the system can quickly detect and parse them, providing the necessary semantic context for the subsequent build phases. At the beginning of the build, the system will perform semantic processing on the pipeline records to identify problems within milliseconds. The pass rate, coverage rate and failed use cases in the testing phase will be incorporated into the feature sequence as important inputs for behavior prediction. The container status and service response index are also added to the chain during deployment, enriching the behavioral trajectory of learners.

In this process, the system always intervenes in the DevOps process in a "process embedding" manner. It does not merely receive the final result but continuously makes judgments and updates along with the task chain, transforming the original "execute - wait - result view" mode into a dynamic feedback process of "execute - immediate judgment - dynamic optimization".

3.2. System Operation Practice in Various DevOps Learning Tasks

Due to the diverse and hierarchical characteristics of the learning content of DevOps, the system processing methods vary in different task scenarios. Take code repair tasks as an example. Such tasks often involve changes in the code's syntax structure. The system will handle the syntax structure part of the code through an automated approach in actual tasks and convert some of the semantics into formal input that the model can read. Building repair tasks focuses on the continuous generation of log information. Therefore, the system needs to promptly capture the fault stack, execution path, and warning sections during the log writing process, so that the model can quickly identify the problem. Take the test improvement task as an example. Mainly based on the qualitative metrics generated by the test, the system will use this method to map the test results to the capability vector and save records of the test process. Taking deployment optimization tasks as an example, the focus is on the use of performance and the formatting of configuration files. The system tracks deployment behavior by analyzing the monitored information and explaining configuration relationships. The processing and input-output structures of the various task types mentioned above are all correlated, as shown in Table 2 specifically.

Table 2. The input and output structure of the system in different DevOps tasks.

Test the type of task	Measure the main input of the system	Measure the internal processing of the system	Measure the feedback output form
Code modification	Measure the difference text and submit the record	Test AST conversion and semantic extraction	Test structured prompts and modification suggestions
Build repair	Test build log	Error test stack parsing	Check the reasons for the failure and the repair path
Test enhancement	Measure coverage and failed use cases	Analysis of measurement indicators and update of behavioral sequences	Suggestions for intensive testing

Test deployment and optimization	Measure the status of the container and response delay	Identification of test performance characteristics	Guidance on parameter tuning for measurement

In actual operation, the system automatically switches the required parsing methods and model links based on the task structure, ensuring that the learning process does not rely on manual judgment and thus guaranteeing the self-consistency and continuity of the task execution chain at the technical level.

3.3. Learning Interaction and Feedback Presentation during System Usage

The focus of DevOps education is to enable learners to communicate with the system in real time through a series of operations. Once the system starts up and encounters a special event (submission, compilation, testing or deployment) during its operation, the system can immediately start to simulate and provide a response. Because this form of deduction can record the information of environmental changes, learners can obtain a technical explanation highly consistent with their current state when receiving feedback, thereby solving the "feedback break" phenomenon in traditional education. The system feedback is structured, including error location pointing, important part flags, dependency predictions, configuration file difference prompts, as shown in Table 3 below.

Table 3. The corresponding structure of interaction events - reasoning mechanisms - feedback types of the system in practice.

Test the types of interaction events	**Test the internal reasoning and processing logic of the system**	**Feedback presentation form**	**Learning status update method**
Test code submission event (Git Push)	Measure and parse code differences, perform semantic extraction, and update task context	Structured code prompts, potential risk point identification	Update the local weights of the code capability vector ccod
Test Build Execution Event (CI Build)	Perform semantic slicing on the streaming build logs and identify error patterns	Explanation of the reasons for build failure and annotation of key log paragraphs	Update the build capability cbuild and the behavior sequence model
Test the Test Run event	Analyze the pass rate, coverage rate and failed nodes, and evaluate the depth of the test	Test improvement suggestions and key use case reminders	Update the testing capability ctest and adjust the capability increment
Test deployment and launch Event (Deploy)	Analyze container status, performance metrics, and abnormal responses	Deployment configuration suggestions and parameter optimization directions	Update the deployment capabilities of cdeploy and monitoring capabilities
Operation Monitoring events (Monitoring)	Continuously track the performance sequence and identify stability trends	Structured reports containing performance bottleneck analysis	Update the monitoring capability cmonitor and status vector

3.4. Organization and Operation of the DevOps Learning Process Supported by the System

During the DevOps training process, the role of the intelligent mentor system is not only reflected in providing support for specific tasks, but also in participating in the management and optimization of the overall learning process. The system will generate records of trainees' behaviors during operation and form a traceable timeline structure based on various actions and behaviors of trainees, as well as related log information, indicators, conditions and other data, serving as the basis for subsequent simulations. In terms of the task execution chain, the system automatically organizes the operation sequence of learners based on the logical dependencies among tasks, making the learning process naturally present a phased trajectory from submission, construction to testing and deployment, ensuring that the task chain remains consistent with the engineering process. For the skill improvement stage, the results of each task are transformed into the added value of skill vectors, enabling the trainees' ability representations to gradually change as their operations evolve. In this way, the teaching methods can adapt to the changes in the context.

Through the collaborative construction of the behavior record chain, execution chain and capability update chain, this system has achieved the continuity and traceability of DevOps learning. With the help of technology, it has fixed the practical action process and formed a clear and sustainable teaching structure, helping practitioners consolidate their learning experience in DevOps practice.

4. Conclusion

This study proposes an AI-driven intelligent mentor system designed to address the real-time, fine-grained instructional needs of DevOps education. By integrating code repository traces, pipeline execution logs, and operational monitoring data, the system overcomes the limitations of traditional teaching methods that often fail to provide timely and context-aligned feedback. The proposed architecture-comprising semantic analysis mechanisms, structured feedback generation, and capability-vector updates-enables the system to accurately interpret learner actions, reconstruct task-chain execution states, and deliver explanations that correspond directly to the learner's operational context.

Through the coordinated construction of the behavior record chain, the execution dependency chain, and the capability update chain, the system achieves continuity, traceability, and adaptivity throughout the learning process. This allows DevOps training to align more closely with real engineering workflows, while ensuring that learners' skills evolve dynamically as they engage in repeated cycles of writing, building, testing, deployment, and monitoring.

Overall, this research provides a practical and theoretically grounded reference framework for developing intelligent support systems in DevOps education. It demonstrates how AI technologies-when coupled with multi-source data reasoning and process-aware modeling-can significantly enhance the immediacy, accuracy, and pedagogical effectiveness of DevOps learning environments.

Reference

1. B. Jadhav, K. Hasija, R. Laharia, and A. MS, "A Comprehensive Framework for Implementing DevOps in Cloud for Education," In *Proceedings of the International Conference on Smart Data Intelligence (ICSMDI 2021).*, May, 2021.
2. H. Ragnarsdóttir, "Orafori islenskra barna fra 4 til 8 ara aldurs: Langtimarannsokn a vaxtarhraa og stougleika," 2018. doi: 10.24270/netla.2018.15
3. T. D. Allen, "Mentoring relationships from the perspective of the mentor," *The handbook of mentoring at work: Theory, research, and practice*, pp. 123-147, 2007.