International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 07 | Jul 2026 www.irjet.net p-ISSN: 2395-0072

AI-Assisted Software Engineering: Measuring Performance, Efficiency and Collaboration

Inapanuri Pravallika[1], Dr. M. Chandra Mohan[2]

[1]M. Tech Scholar, Dept. Of CSE, JNTUH UCESTH, Hyderabad, India
[2]Professor, Dept. Of CSE, JNTUH UCESTH, Hyderabad, India

Abstract - *Artificial Intelligence (AI) is transforming software engineering by assisting developers in writing, debugging, and improving code. However, most existing AI coding assistants require an internet connection and primarily focus on code generation without evaluating their overall impact on software development. This paper presents an offline AI-assisted software engineering system that supports programmers by detecting and correcting common Python programming errors while maintaining human control over AI-generated suggestions. The proposed system enables users to execute code, receive AI-based corrections, and choose whether to accept or reject the suggested changes. All user interactions are recorded and analyzed to evaluate the effectiveness of AI assistance. The system measures software performance using Success Rate, Error Rate, and Error Reduction Rate, development efficiency using Average Execution Time, and human–AI collaboration using AI Usage Rate, AI Acceptance Rate, Code Similarity, Average Lines Changed, and the AI Collaboration Index (ACI). Experimental results demonstrate that the proposed framework effectively reduces programming errors, improves development efficiency, and provides a quantitative approach for evaluating human–AI collaboration in software engineering. The proposed system offers a practical, privacy-preserving, and scalable solution for AI-assisted programming in offline environments.*

Key Words: Artificial Intelligence, Software Engineering, Human–AI Collaboration, Performance Measurement, Efficiency Analysis, AI Collaboration Index, Offline Coding Assistant.

1. INTRODUCTION

Artificial Intelligence (AI) has emerged as a transformative technology in software engineering by assisting developers in various programming activities, including code generation, error detection, debugging, and code completion. AI-assisted coding tools have become increasingly popular because they help improve software quality, reduce manual effort, and accelerate the software development process. By automating repetitive programming tasks and providing intelligent code suggestions, AI enables developers to focus on solving complex problems and improving application quality. As a result, AI-assisted software development has become an important area of research in modern software engineering.

Despite these advancements, most existing AI-assisted coding tools are primarily designed to generate code or provide debugging support. These systems often depend on cloud-based services, requiring continuous internet connectivity and raising concerns regarding data privacy, security, and accessibility. Moreover, while current AI tools assist developers during programming, they rarely provide a comprehensive framework to evaluate the overall effectiveness of AI assistance. In particular, there is limited research on quantitatively measuring how AI influences software performance, development efficiency, and human–AI collaboration during the software development lifecycle.

The growing integration of AI into software engineering highlights the need for an objective evaluation framework that measures not only the quality of AI-generated code but also its contribution to developer productivity and collaboration. Such a framework can help researchers and developers understand the effectiveness of AI-assisted programming and support the development of more reliable and efficient intelligent coding systems. An offline AI-assisted environment further addresses concerns related to internet dependency and data confidentiality while providing consistent support for programmers.

To address these challenges, this paper proposes an offline AI-Assisted Software Engineering framework that assists programmers in detecting and correcting common Python programming errors using a rule-based AI assistant. The proposed system enables users to execute programs, receive AI-generated code suggestions, and decide whether to accept or reject the corrections. In addition to providing coding assistance, the system records user interactions and evaluates AI effectiveness using quantitative metrics. Software performance is measured through Success Rate, Error Rate, and Error Reduction Rate. Development efficiency is evaluated using Average Execution Time, while human–AI collaboration is assessed using AI Usage Rate, AI Acceptance Rate, Code Similarity, Average Lines Changed, and the AI Collaboration Index (ACI).

The proposed framework not only supports developers in improving code quality but also provides a structured

methodology for evaluating the impact of AI assistance on software development. By combining offline AI-assisted debugging with quantitative performance analysis, the system offers a practical and privacy-preserving solution for assessing the effectiveness of AI-assisted software engineering. The experimental results demonstrate that the proposed framework effectively enhances the coding process while maintaining active human involvement in decision-making.

1.1 PROBLEM STATEMENT

The increasing adoption of Artificial Intelligence (AI) in software engineering has significantly improved code generation, debugging, and programming assistance. However, most existing AI-assisted coding tools primarily focus on providing code suggestions and error correction without offering a comprehensive framework to evaluate their impact on software development. In particular, they do not quantitatively measure key aspects such as software performance, development efficiency, and human–AI collaboration. Additionally, many AI coding assistants rely on cloud-based services, requiring continuous internet connectivity and raising concerns related to data privacy, security, and accessibility.

To address these limitations, there is a need for an offline AI-assisted software engineering system that not only supports developers in identifying and correcting programming errors but also provides measurable insights into the effectiveness of AI assistance. Therefore, this research proposes an offline AI-assisted coding environment that evaluates AI support using quantitative metrics, including Success Rate, Error Rate, Error Reduction Rate, Average Execution Time, AI Usage Rate, AI Acceptance Rate, Code Similarity, and the AI Collaboration Index (ACI). These metrics enable an objective assessment of AI's contribution to software performance, development efficiency, and human–AI collaboration.

2. LITERATURE SURVEY

Artificial Intelligence (AI) has significantly transformed software engineering by providing intelligent assistance in code generation, debugging, and software maintenance. Modern AI-assisted coding tools such as GitHub Copilot, Code Whisperer, and other intelligent programming assistants help developers improve productivity by offering code completion, error detection, and automated code suggestions. Recent studies have shown that these tools can reduce development time and improve programming efficiency by assisting developers in repetitive coding tasks. However, most existing AI-assisted systems primarily focus on generating code and correcting syntax errors rather than evaluating the overall impact of AI on software development.

Several researchers have investigated the use of machine learning and rule-based techniques to support software development through automated debugging and intelligent code recommendations. These approaches have demonstrated improvements in code quality and reduced debugging effort by identifying common programming mistakes and suggesting appropriate corrections. Although these methods enhance the coding process, many of them depend on cloud-based AI services, requiring continuous internet connectivity. This dependency raises concerns regarding data privacy, security, and accessibility, especially in environments where offline development is preferred. Furthermore, existing studies rarely provide quantitative measures to evaluate how AI assistance influences software performance, development efficiency, and collaboration between developers and AI systems.

Based on the reviewed literature, it is evident that a comprehensive evaluation framework for AI-assisted software engineering is still lacking. Most existing solutions emphasize AI-generated code without systematically measuring its effectiveness in real-world software development. To address this research gap, the proposed work presents an offline AI-assisted software engineering framework that combines rule-based error correction with quantitative evaluation metrics. The system measures software performance using Success Rate, Error Rate, and Error Reduction Rate, development efficiency using Average Execution Time, and human–AI collaboration using AI Usage Rate, AI Acceptance Rate, Code Similarity, Average Lines Changed, and the AI Collaboration Index (ACI). This integrated approach provides a structured and objective framework for evaluating the effectiveness of AI-assisted software engineering while maintaining active human involvement throughout the development process.

3. PROPOSED METHODOLOGY & DESIGN

The proposed AI-Assisted Software Engineering system provides an offline environment that supports programmers in writing, executing, and debugging Python programs while evaluating the effectiveness of AI assistance. The system is implemented using the Flask framework and follows a modular architecture consisting of the User Interface, Code Execution, AI Assistance, Logging, Analysis, and Results modules. The overall workflow enables users to interact with the AI assistant while collecting data required to evaluate software performance, development efficiency, and human–AI collaboration.

3.1 Proposed System Architecture

The proposed architecture consists of five major layers. The User Layer allows programmers to write Python code, execute programs, request AI suggestions, accept or reject AI-generated corrections, and view the

results. The Web Application Layer manages user interactions through the Flask framework, executes the submitted code, detects programming errors, and generates rule-based AI suggestions for correction.

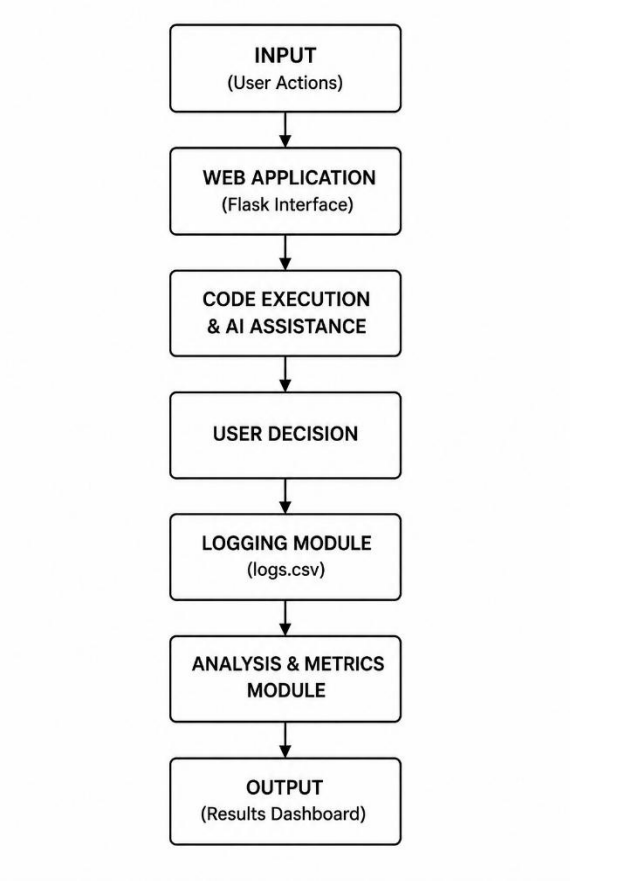

Fig-1: system architecture

The Logging Layer records all user interactions, including execution status, AI usage, AI acceptance, error information, execution time, code similarity, and lines changed. These records are stored in a CSV log file, which serves as the dataset for further analysis.

The Analysis and Metrics Layer processes the collected log data and computes software performance, development efficiency, and human–AI collaboration metrics. Performance is evaluated using Success Rate, Error Rate, and Error Reduction Rate. Efficiency is measured using Average Execution Time, while collaboration is assessed using AI Usage Rate, AI Acceptance Rate, Code Similarity, Average Lines Changed, and the AI Collaboration Index (ACI).

Finally, the Output Layer presents the computed results through a dashboard, enabling users to visualize the effectiveness of AI-assisted software development.

3.2 System Workflow

The workflow begins when the user enters Python source code through the web interface. The Code Execution Module executes the program and checks for errors. If the program executes successfully, the output is displayed and the execution details are logged. If an error occurs, the AI Assistance Module analyzes the error and generates a corrected version of the code using predefined rule-based techniques. The user can then accept or reject the AI suggestion. When the suggestion is accepted, the corrected code is executed again and the system calculates code similarity and the number of modified lines. All interaction details are stored in the log file and later processed to compute performance, efficiency, and collaboration metrics. The final results are displayed on the dashboard for analysis.

3.3 Performance Measurement

1. Success Rate (SR):

The Success Rate indicates the percentage of programs executed successfully without any errors.

$$\text{SR} = \frac{\text{Successful Executions}}{\text{Total Executions}} \times 100$$

2. Error Rate (ER):

The Error Rate measures the proportion of executions that resulted in errors.

$$\text{ER} = \frac{\text{Executions with Errors}}{\text{Total Executions}}$$

3. Error Reduction Rate (ERR):

This metric evaluates the effectiveness of AI in reducing coding errors.

$$\text{ERR} = \frac{\text{Errors}_{\text{before}} - \text{Errors}_{\text{after}}}{\text{Errors}_{\text{before}}} \times 100$$

3.4 Efficiency Measurement

1. Average Execution Time (AET):

$$\text{AET} = \frac{\sum \text{Execution Time}}{\text{Total Executions}}$$

3.5 Human–AI Collaboration Measurement

1. AI Usage Rate (AIUR):

$$\text{AIUR} = \frac{\text{AI Used}}{\text{Total Executions}} \times 100$$

2. AI Acceptance Rate (AIAR):

$$\text{AIAR} = \frac{\text{Accepted Suggestions}}{\text{Generated Suggestions}} \times 100$$

3. Code Similarity Score (CSS):

$$\text{CSS} = \frac{\text{Matching Code}}{\text{Total Code}} \times 100$$

3.6 Proposed Algorithm

The proposed algorithm begins by accepting Python source code entered by the user through the web-based interface. The submitted code is executed in a controlled environment, where the system captures the execution output, execution time, and any errors encountered during execution. If the program executes successfully, the output is displayed to the user, and the execution details are recorded in the log file. If an error is detected, the AI Assistance Module analyzes the error and generates a corrected version of the code using predefined rule-based techniques. The corrected code is then presented to the user for review.

The user is provided with the option to either accept or reject the AI-generated suggestion. If the suggestion is accepted, the corrected code replaces the original code and is executed again to verify successful execution. The system then calculates the code similarity between the original and AI-corrected versions and determines the number of lines modified. If the user rejects the suggestion, the original code remains unchanged. All interaction details, including AI usage, AI acceptance, execution status, execution time, code similarity, and lines changed, are stored in a CSV log file. Finally, the logged data is processed to compute software performance metrics (Success Rate, Error Rate, and Error Reduction Rate), efficiency metrics (Average Execution Time), and human–AI collaboration metrics (AI Usage Rate, AI Acceptance Rate, Code Similarity, and Average Lines Changed). The computed results are displayed through the dashboard, providing an objective evaluation of the effectiveness of AI-assisted software engineering.

4. RESULTS

Figure 2 shows the user interface developed for the proposed system. It allows users to write and execute Python programs, request AI-assisted code corrections, and accept or reject AI-generated suggestions. The interface provides immediate feedback by displaying execution results and error messages, enabling effective interaction between the programmer and the AI assistant.

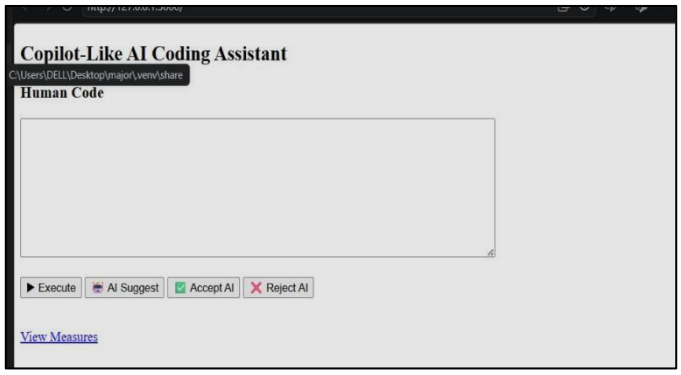

Fig -2: User interface

Figure 3 presents the developer interaction log generated by the proposed system. The log records important information such as the user action, AI usage, AI acceptance, execution status, execution time, code similarity, and the number of lines changed. These records serve as the experimental dataset for evaluating the performance, efficiency, and collaboration metrics.

Fig -3: Developer interaction log

Figure 4 shows the AI interaction log that captures the AI-assisted coding activities during program execution. The logged information is used to calculate AI Usage Rate, AI Acceptance Rate, Code Similarity, Average Lines Changed, and the AI Collaboration Index (ACI). This log enables a quantitative analysis of human–AI collaboration within the proposed system.

Fig -4: AI interaction log

PERFORMANCE RESULTS:

The performance results presented in Table 1 demonstrate the effectiveness of the proposed AI-assisted software engineering system in improving code execution. The Success Rate represents the percentage of programs executed successfully, while the Error Rate indicates the proportion of executions that resulted in errors. The Error Reduction Rate reflects the system's ability to minimize programming errors through AI-assisted code correction. These results indicate that the proposed framework effectively supports developers by reducing coding errors and improving overall software performance.

Table 1: Performance results

Metric	Value
Success Rate	51.39
Error Rate	48.61%
Error Reduction Rate	51.39

EFFICIENCY RESULTS:

Table 2 presents the efficiency evaluation of the proposed system using the Average Execution Time. This metric represents the average time required to execute user programs during the experimental evaluation. A lower execution time indicates better system efficiency and faster program execution. The obtained results show that the proposed framework provides timely feedback and supports efficient debugging while maintaining a smooth coding experience.

Table 2: Efficiency results

Metric	Value
Average Execution Time	30sec

HUMAN-AI COLLABORATION RESULTS:

Figure 5 presents the human–AI collaboration results obtained from the experimental evaluation of the proposed system. The collaboration analysis is performed using metrics such as AI Usage Rate, AI Acceptance Rate, Code Similarity, Average Lines Changed, and the AI Collaboration Index (ACI). These metrics are computed from the interaction logs generated during the coding process and provide a quantitative assessment of the collaboration between the programmer and the AI assistant.

The results indicate that the AI assistant effectively supports programmers by generating useful code corrections while allowing users to decide whether to accept or reject the suggestions. The calculated collaboration metrics demonstrate the level of AI involvement, user trust in AI-generated solutions, and the extent of code modifications introduced by the AI. Overall, the findings show that the proposed framework promotes effective human–AI collaboration while ensuring that the programmer retains control over the final software development process.

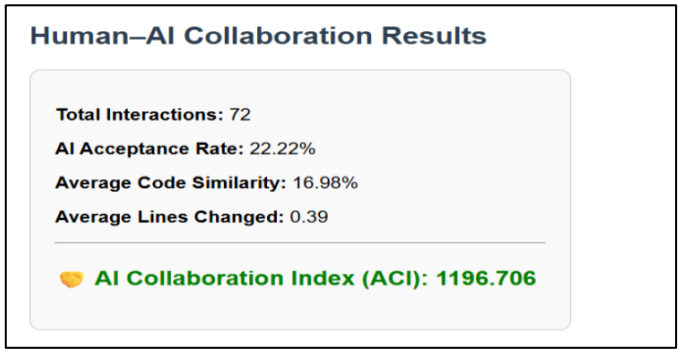

Fig -5: human-AI collaboration results

CONCLUSION:

This paper presented an offline AI-assisted software engineering framework designed to support programmers in code execution, error detection, and automated code correction while providing a quantitative evaluation of AI-assisted software development. The proposed system integrates a rule-based AI assistant with a modular architecture to assist developers in correcting common Python programming errors without requiring an internet connection. In addition to providing coding assistance, the system evaluates its effectiveness using measurable metrics related to software performance, development efficiency, and human–AI collaboration. Performance was assessed using Success Rate, Error Rate, and Error Reduction Rate, while efficiency was evaluated through Average Execution Time. Human–AI collaboration was measured using AI Usage Rate, AI Acceptance Rate, Code Similarity, Average Lines Changed, and the AI Collaboration Index (ACI). The experimental results demonstrate that the proposed framework effectively reduces programming errors, provides timely debugging support, and promotes meaningful collaboration between developers and AI while maintaining human control over coding decisions. The proposed approach offers a simple, privacy-preserving, and practical solution for evaluating AI-assisted software engineering and provides a foundation for future research on intelligent software development environments.

REFERENCES:

1) Kumar, P. Singh and S. Bhattacharya, "A Survey on AI-Assisted Code Generation Tools and Their Impact on Software Development," 2023 International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 156-162, June 2023.

2) C.Hill,F.Liu and X. Zhang, "Evaluating Human-AI Collaboration in Code Completion Tools," IEEE Transactions on Software Engineering, vol. 49, no. 4, pp. 987-1001, Apr. 2023.

3) M. Allamanis, E. T. Barr, P. Devanbu and C. Sutton, "A Survey of Machine Learning for Big Code and aturalness," ACM Computing Surveys, vol. 51, no. 4, pp. 1-37, July 2018.

4) G. Cupic, N. Stojanovic and M. Vukovic, "An Intelligent Code Assistance Tool for Automatic Bug Fixing," 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO), pp. 368-373, May 2021.

5) H. Li, Y. Wang and Q. Zou, "A Framework for Measuring AI Collaboration in Software Development," 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 455-466, Feb. 2022.

6) J. Zhang and B. Li, "Human–AI Interaction Models for ode Generation Tasks," IEEE Software, vol. 38, no.2, pp. 78-85, Mar./Apr. 2021.

7) S. Raychev, M. T. Vechev and A. S. Bielik, "Patch Prediction for Error Correction Using Statistical Learning," Proc. ACM on Programming Languages, vol. 2, OOPSLA, pp. 1-26, Nov. 2018.