คณะวิศวกรรมศาสตร์
Faculty of Engineering

PROJECT DAY 2026
FACULTY OF ENGINEERING

INC07

Source Code Similarity Detection System in a Programming Contest Platform

CONTROL SYSTEMS AND INSTRUMENTTAION ENGINEERING PROGRAM

Wongsatorn Saengain Advisor Assoc Prof.DrPoj Tangamchit

Introduction

Automated platforms like **DOMjudge** efficiently grade functionality but cannot detect **code similarity.** Manual inspection of modified code is slow and error-prone. This project proposes an automated logic-level detection system using **Abstract Syntax Trees** and **Tree Edit Distance** to evaluate true structure while ignoring superficial changes, reducing workload and supporting academic integrity.

Method

This study develops a logic-level source code similarity detection system integrated with the DOMjudge platform. The overall workflow

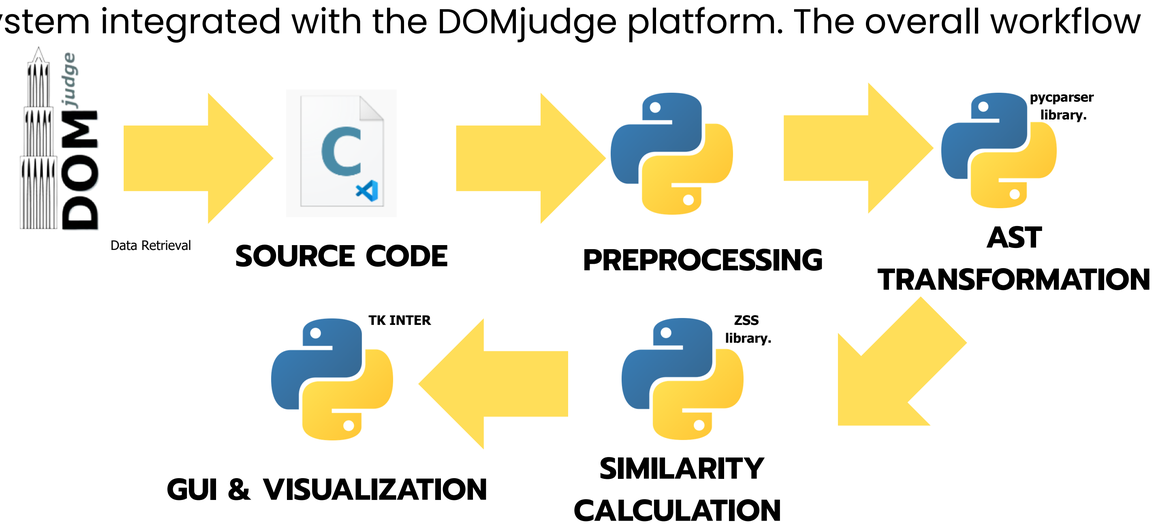


Fig. 1. System architecture and workflow of the proposed source code similarity detection system.

Data Retrieval	Fetch data directly via the DOMjudge API.
Preprocessing	Removes directives and comments pycparser cannot handle them.
AST Transformation	Convert code into an Abstract Syntax Tree using the pycparser library.
Similarity Calculation	Uses Tree Edit Distance to measure structural differences between code submissions.
Matrix Export	Output the all-pairs similarity scores into a structured CSV matrix file for data persistence
GUI Visualization	Tkinter GUI for data visualization, similarity detection, and evaluation workflows

Theory

Fig 2. Abstract syntax tree

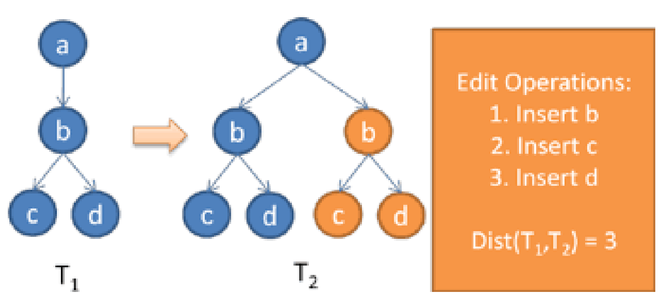
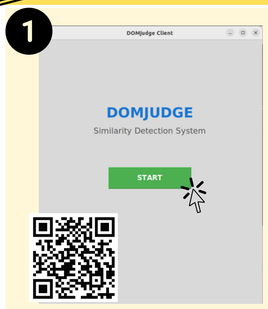


Fig 3. Tree edit distance between T_1 and T_2

Abstract syntax tree	A structural representation of source code based on its syntax
Tree Edit Distance	A measure of similarity between two trees based on edit operations

Results

1



GUI INTERFACE

The GUI integrates with the server to monitor submissions download student files and perform similarity analysis in a single workflow.

2

Team	1201	1202	1203	1204	1205
1201	100	97.87	100	95.83	56.41
1202	97.87	100	97.87	97.92	55.13
1203	100	97.87	100	95.83	56.41
1204	95.83	97.92	95.83	100	53.85
1205	56.41	55.13	56.41	53.85	100

CSV

The CSV file shows similarity scores between each pair of students.

CSV FILE

3

VISUALIZATION

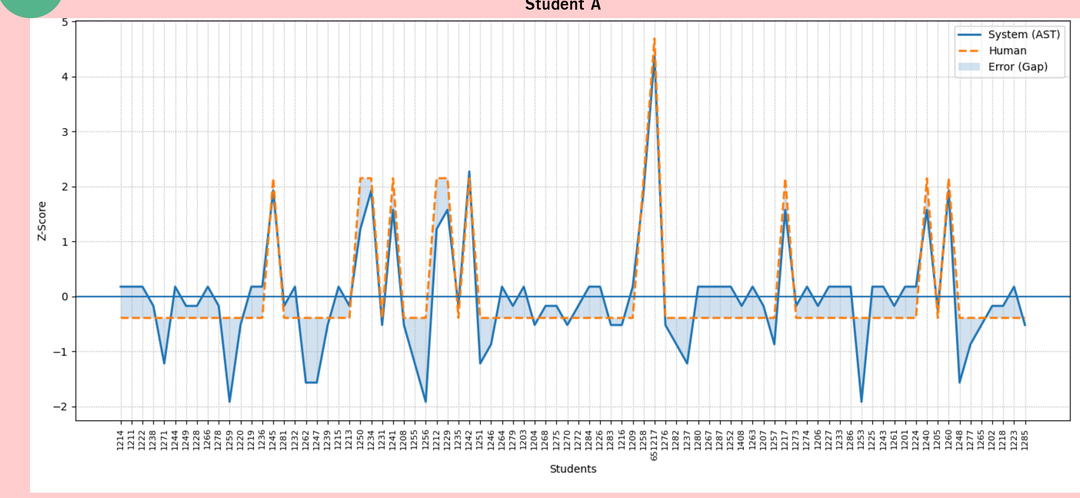


Fig. 4. Student A Low Similarity Case

Student A represents the case with the lowest similarity scores across the dataset, indicating minimal resemblance to other students. In this case, both system and human evaluations remain consistently low, demonstrating that the system correctly identifies low-similarity patterns.

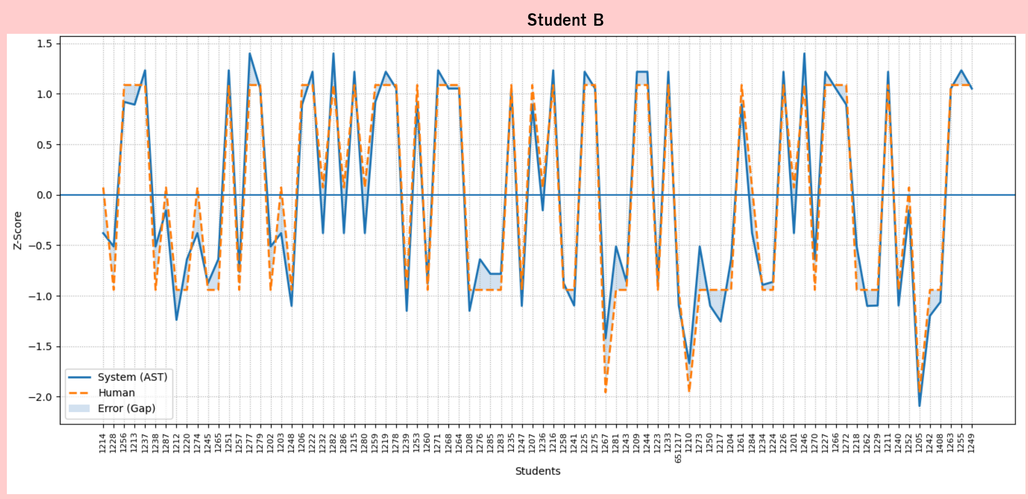


Fig 5. Student B High Similarity Case

Student B represents the case with the highest similarity scores, indicating strong resemblance with multiple submissions. The system-generated scores closely follow the human evaluation trend, even in high-similarity regions.

Conclusion

The system closely matches human judgment with consistent performance and low error across different similarity scenarios. patterns while reducing the workload of manual code inspection through automated analysis.

The system provides a reliable, efficient, and scalable solution for large-scale code similarity evaluation.

reference

I. D. Baxter et al., "Clone detection using abstract syntax trees," in Proc. Int. Conf. Softw. Eng. (ICSE), 1998, pp. 368-377. DOI: 10.1109/ICSE.1998.671138