Software Testing by Metrics Calculation

Ms. R. Hinduja[1], V. Akash[2], and A. Arshath [3]
[1] *Department of Software System, Sri Krishna Arts and Science College*
[2,3] *PG student of Computer Science, Sri Krishna Arts and Science College*

Abstract: **Software testing plays an indispensable role in ensuring the quality, reliability, and maintainability of applications. This paper introduces a system designed to automate the calculation of software metrics for Python projects. By analyzing key metrics such as lines of code (LOC), number of classes, methods, Response for Class (RFC), Coupling Between Object Classes (CBO), and Lack of Cohesion in Methods (LCOM), the system provides quantitative insights into code quality. The results are stored in an Excel file for further analysis, allowing developers to assess the maintainability and design quality of their software. This automation significantly reduces manual effort and error while offering a robust tool for software quality assurance.**

I. INTRODUCTION

Software quality assurance is a critical aspect of the software development lifecycle. With increasing complexity in modern software, manual testing methods are no longer sufficient to evaluate quality comprehensively. Metrics-based analysis provides developers with a quantitative understanding of their code's maintainability, complexity, and reliability. However, traditional tools often lack integration with modern development workflows, especially for Python projects. This paper presents a Python-based system to automate the calculation of essential software metrics, addressing this gap. The system provides actionable insights into code quality, enabling developers to identify design flaws early in the development process.

II. DESCRIPTION

The proposed system is developed using Python, featuring a user-friendly GUI designed with PyQt5. It automates the process of metrics calculation by parsing the selected project directory, analysing Python files, and computing critical metrics. These metrics include measures of coupling, cohesion, and complexity, providing a holistic view of the project's quality. The system seamlessly integrates with Python-based workflows, allowing results to be exported to an Excel file for further review and reporting. This automation reduces manual effort while ensuring accuracy and scalability.

III. DATASET COLLECTION

To validate the system, various open-source Python projects were used as datasets. These projects, varying in size and complexity, were selected to test the system's versatility and robustness. The datasets provided diverse scenarios, including small scripts, medium-sized applications, and large-scale frameworks. This diversity ensured the system could handle real-world challenges in analysing Python code and computing metrics efficiently.

IV. EXISTING SYSTEM

Existing systems for software metrics calculation often rely on standalone tools or manual analysis, which are inefficient and error-prone. These systems generally lack support for Python-specific features and fail to provide automated solutions for storing results. Moreover, traditional methods do not integrate seamlessly into modern development workflows, limiting their usability. The need for a Python-specific, automated, and user-friendly tool has driven the development of the proposed system.

V. PROPOSED SYSTEM

The proposed system addresses the limitations of existing methods by automating the process of metrics calculation for Python projects. It allows users to select a project directory and computes metrics such as lines of code, number of classes, number of methods, RFC, CBO, LCOM, and others. The results are displayed in a GUI and exported to an Excel file for further analysis. This system improves efficiency, accuracy, and integration, making it a valuable tool for Python developers to evaluate code quality and maintainability.

VI. LITERATURE REVIEW

Software metrics have long been recognized as critical indicators of code quality and maintainability. Research highlights the importance of cohesion and coupling metrics in predicting fault-proneness and

maintainability. Studies emphasize that high coupling and low cohesion often lead to poorly designed systems, increasing maintenance costs and reducing reliability. While several tools exist to compute these metrics, they often focus on Java or C++ projects, with limited support for Python. This paper fills this gap by providing a Python-centric solution for automated metrics calculation.

VII. METRICS CALCULATION

The system calculates a range of metrics that provide insights into various aspects of software quality:

- Lines of Code (LOC): Measures the size of the project and is a basic indicator of complexity.
- Number of Classes and Methods: Indicates modularity and abstraction.
- Response for Class (RFC): Measures the complexity of a class by counting the number of methods that can be invoked in response to a message.
- Coupling Between Object Classes (CBO): Reflects dependencies between classes, where higher values indicate potential design issues.
- Lack of Cohesion in Methods (LCOM): Measures the degree to which methods in a class are related, with higher values indicating poor cohesion.
- Depth of Inheritance Tree (DIT): Reflects inheritance complexity and its impact on code readability and maintainability.

VIII. WORKFLOW

The system operates in a structured workflow:

1. The user selects a Python project directory via the GUI.
2. The Python ast library parses the files to extract relevant information about classes, methods, and code structure.
3. The system computes metrics using algorithms tailored for Python’s object-oriented features.
4. Results are displayed in a user-friendly interface and exported to an Excel file for detailed analysis and reporting.
5. Developers analyze the metrics to assess design quality and identify potential areas for improvement.

XI. RESULTS

The system was tested on various open-source Python repositories. The metrics provided insights into the quality of the projects, identifying areas where cohesion was low or coupling was high. The ability to export results to Excel streamlined reporting and facilitated further analysis. Developers could quickly identify problematic areas, such as deeply nested inheritance trees or highly coupled classes, and address them to improve code maintainability and reliability.

X. CONCLUSION

This paper presents an efficient system for automated software metrics calculation in Python projects. By focusing on key metrics such as cohesion, coupling, and complexity, the system provides actionable insights into code quality. The integration of a GUI and Excel export functionality enhances usability and efficiency, making it a valuable tool for software developers. Future work will focus on extending the system to support additional metrics and integrate it with continuous integration pipelines for real-time quality assessment.

REFERENCES

[1] Pressman, R. S., *Software Engineering: A Practitioner's Approach*, McGraw Hill.
[2] Chidamber, S. R., & Kemerer, C. F., "A Metrics Suite for Object-Oriented Design," *IEEE Transactions on Software Engineering*.
[3] Fenton, N., & Pfleeger, S. L., *Software Metrics: A Rigorous and Practical Approach*.