AUTOMATED GRADING SYSTEM USING STATIC CODE ANALYSIS TO ELIMINATE MANUAL GRADING PROCESS FOR PROGRAMMING ASSIGNMENTS

BACHELOR ESSAY PROPOSAL
(PROPOSAL SKRIPSI)
2026

By:
Dicha Zelianivan Arkana NIM. 2241720002

INFORMATICS ENGINEERING STUDY PROGRAM
INFORMATION TECHNOLOGY DEPARTMENT
STATE POLYTECHNIC OF MALANG
2026

Table of Contents

1. Introduction

1.1. Background

In recent times, many aspects of our life have started to be automated. Although, there are still many aspects that is left to be done by humans, especially in the field of education. One of the aspects is the process of grading assignments. The significant growth of the computer science course over the years has lead to more assignments to grade (Krusche et al., 2020).

Unfortunately, it is often impossible to personally review students' code and provide prompt, thorough comments, and professional static analysis tools are inappropriate for use in educational settings.. This is because the grading process is often manual and subjective. The chance of inconsistency in grading accuracy, feedback quality, and the overall variance in the grading process is high when done by humans (Messer et al., 2024).

There has been several studies attempting to automate the grading process of programming assignments, but most of them have been limited to assessing the correctness of the solution instead of how the actual solution is written (Messer et al., 2024).

Static code analysis usually refers to the analysis of a computer program without executing it. This is usually done by analysing the source code of the program and using several heuristics to determine the correctness of the program, whether it be stylistic, syntax errors, or possible logical errors.

This study will explore the use of static code analysis to automatically grade programming assignments as opposed to using it as a tool to check for correctness. This study will also discuss the benefits and drawbacks of using such method to grade assignments.

1.2. Problem Statement

There is a clear gap in regards to the grading process of programming assignments. This study will try to answer the following questions:

1. How effective is the use of static code analysis to grade programming assignments?
2. What are the benefits of using static code analysis to grade assignments?
3. How can static code analysis be used to grade programming assignments?
4. What are the drawbacks of using static code analysis to grade programming assignments?
5. What are the limitations of using static code analysis and how to improve them?

1.3. Objective of the study

The main objective of this study is to explore how viable it is to use static code analysis to automatically grade programming assignments. Not limited to that, this study have other related objectives which are listed below:

1. To assess the effectiveness of doing automated grading of programming assignments is more effective that its manual counterpart.
2. To investigate whether or not using static code analysis to grade programming assignments is the viable and effective way to grade assignments.
3. To identify the impact of automated grading process to the students' learning process and how it makes the lecturer more time-efficient.

4. To identify if there are any drawbacks of using static code analysis to grade assignments and how to overcome them.

1.4. Significance of the study

This study will try to fill in the gaps in the current grading process of programming assignments by using static code analysis. It will focus on the correctness of how the solution is written and how the solution is structured as opposed to the actual execution response of the solution.

1.5. Scope of the study

The method used in this study will not involve the use of either human graders or artificial intelligence to grade the assignments. Instead, it will focus on using static code analysis to grade the assignments to provide the students with reproducible and consistent feedback. This study will not involve checking the correctness of the solution since it requires executing the solution, which prevents the student getting and instant feedback and quick iteration workflow to improve their solution.

There will also be data collection and analysis to evaluate the effectiveness of this method, though it will not be a primary focus of this study.

2. Review of Related Literature

2.1. Previous Research Finding

The previous research that was done by (Krusche et al., 2020) states that in recent years, there has been a significant increase in the number of programming assignments in computer science courses with varying degrees of difficulty. (Krusche et al., 2020) proposes a method to improve the engagement of students in modelling by using an interactive learning method supported by a software. The grading process is integrated in the software and automatically done by the software. The software is able to provide instant feedback to the students.

The study conducted by (Syaifudin et al., 2024) proposes a method to automate the grading process of programming assignments for android applications using unit testing and integration testing. Compared to (Krusche et al., 2020), this method is more detached from the application used to write the assignments while still being able to provide feedback to the students.

Several other tools has been proposed to automate the grading process as seen in (Rajesh et al., 2024). These tools has evolved over the years and have been used in various educational settings. However, there is still not a clear consensus on which tools are the most effective and why.

GradeStyle by (Iddon et al., 2023) is a tool that integrates with Github and provides automated feedback each time a student pushes their code to the repository. This making the process of grading assignments more efficient because most students already have Github account and can easily push their code to the repository and get feedback.

2.2. Foundational Framework

The foundational framework of this study is the use of static code analysis to grade programming assignments. This study specifically focuses more on the correctness of how the solution is written and how the solution is structured as opposed to the actual execution response of the

solution. Although, even if the main objective of this study is to try to score the correctness of how the code is structured, it is important to note that the result of the execution response of the solution is also considered albeit with a lower percentage.

3. Research Methodology

Grading programming assignments often involves a subjective process. Most of the time, it only focuses on the correctness of the solution and not the correctness of how the solution is written. The main reason of this study is to build a system that can automatically grade programming assignments mainly based on the correctness of how the solution is written utilising static code analysis, while also taking into account the correctness of the code execution.

This study will focus more on the qualitative aspects of the assignments rather than the quantitative aspects. This is because the current focus of the study is to see whether or not the use of static code analysis is feasible to grade programming assignments. The quantitative aspects of the assignments will be evaluated later in the study.

3.1. Data Collection

4. Research Finding and Discussion

5. Conclusion and Suggestion

5.1. Conclusions

5.2. Suggestions

References

Iddon, C., Giacaman, N., & Terragni, V. (2023). GRADESTYLE: GitHub-Integrated and Automated Assessment of Java Code Style. *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 192–197. https://doi.org/10.1109/ICSE-SEET58685.2023.00024

Krusche, S., Frankenberg, N. von, Reimer, L. M., & Bruegge, B. (2020). An interactive learning method to engage students in modeling. *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training*, 12–22. https://doi.org/10.1145/3377814.3381701

Messer, M., Brown, N. C., Kölling, M., & Shi, M. (2024). Automated Grading and Feedback Tools for Programming Education: A Systematic Review. *ACM Transactions on Computing Education*, *24*(1). https://doi.org/10.1145/3636515

Rajesh, S., Rao, V. V., & Thushara, M. (2024). Comprehensive Investigation of Code Assessment Tools in Programming Courses. *2024 IEEE 9th International Conference for Convergence in Technology (I2ct)*, 1–6. https://doi.org/10.1109/I2CT61223.2024.10543863

Syaifudin, Y. W., Saputra, P. Y., Fatmawati, T., Susanti, N. I., Patta, A. R., & Calvina, A. (2024). Application of Unit Testing and Integration Testing for Automatic Grading Mechanism in Android Programming Learning Assistance System. *2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)*, 1883–1887. https://doi.org/10.1109/ICETSIS61505.2024.10459673