Using SonarQube to Evaluate the Code Quality of C/C++ Programming Assignments

Xiao Liu*, Yeoneo Kim*, Sugwoo Byun**, Gyun Woo*,***
*Dept. of Electrical and Computer Engineering, Pusan National University
**Department of Computer Engineering, Kyungsung University, Busan, Korea
***Smart Control Center of LG Electronics
e-mail:*{liuxiao, woogyun}@pusan.ac.kr, **swbyun@ks.ac.kr

Abstract

The programming code can be measured in several concepts. One of the critical aspect is the correctness. However, the grading of the code quality is also important and should be evaluated. This paper presents an approach to evaluate the C/C++ programming assignments by applying SonarQube to check the code quality. SonarQube is a wildly used software quality management platform. The accuracy of evaluating the qualities of Java and Python assignments has been proved in our previous work. This paper uses a SonarQube plugin named sonar-cxx to detect the quality of C/C++ based programs. The detected code quality result is combined with its correctness result to generate a final score which is more appropriate to represent the accuracy of a programming assignment.

1. Introduction

The code quality is as much important as its correctness in industries, and it is necessary to educate how to improve the code quality as a student. However, it is rare to find a system which is able to not only determine the correctness of program output but also evaluate its code quality [1].

Our previous work introduced using SonarQube built-in functions to detect the Java and Python based programming assignments to evaluate them more accurately [2]. Since the C/C++ programming is also one of the fundamental skills in computer science field, it is also necessary to apply the code quality detection for the C/C++ programming assignments.

As an open platform, SonarQube manages the code quality during the program development lifecycle. The SonarQube is able to detect code qualities such as finding bugs, checking vulnerabilities, and code smells. SonarQube supports several predefined and customized rules as measures for quality detection [3].

We use neoESPA as an automatic coding judging system for last three years in undergraduate programming courses. It can evaluate the correctness of programming codes written in multiple programming languages [4].

This paper presents an approach to evaluate the C/C++ programming assignments more accurately by using SonarQube and its plugin named sonar-cxx. We apply this approach on neoESPA to make it not only grade the correctness of C/C++ assignments but also detect their qualities.

This paper is organized as follows. Section 2 introduces the background including the mechanism of SonarQube. Section 3 illustrates the development of our proposed work and the experiment result. Section 4 discusses some limitations in our system, and Section 5 concludes.

2. Background

The SonarQube platform and its community help developers to continuously manage the health of their softwares. The SonarQube consists of two parts: a server and a scanner. The SonarQube server provides an environment to mange and execute the data and functions. The SonarQube scanner runs on the server as one of functions to detect code quality.

The quality detection features of SoarQube rely on the principles which summarized based on the programming regulations and de facto patterns used to standardize the programming style. Such principles are regulated into rules used to measure whether a program code is developed well or not. The rules cover several parts of the code quality such as efficiency, robustness, and security [5].

SonarQube also allows users to add their own rules to detect particular programming issues. According to the criticality, a custom rule can be assigned to four levels which are bugs, vulnerabilities, security hotspots, and code smells, respectively. Also, SonarQube allows users to add custom rules to detect the particular mistakes.

SonarQube is capable for detecting the qualities of Java and Python based programs by default. A SonarQube plugin named sonar-cxx is an open source project which is able to detect the code qualities of C/C++ programs.

3. Proposed work

Since the most basic requirement of an assignment code is the processability, i.e., the source code should be successfully compiled and generates an executable, we execute the correctness check first, and then applying the SonarQube functions to detect its code quality.

Table 1. The Custom rules for quality detection

Rule name	Rule description	Weight
complying naming specification	The names of functions should be defined as required in the assignment specification	2
Avoiding using forbidden operators	The particular operation should be implemented by students and without using existing ones.	2
Avoiding the same conditions "if/else if" statement	Only one brach will be executed if an "if/else if" statement contains the same conditions	2

Table 2. The Predefined rules for quality detection

Rule name	Rule description	Weight
Missing curly brace	Not using curly braces could be error-prone	3
too many nested conditional statements	Too many nested conditional statements make Spaghetti code	3
Collapsible if statement	Merging collapsible if statement can increase readability	3

Figure 1 demonstrates the data flow of an assignment submission and assessment procedure in the proposed system. At the beginning, a student submits his/her source code to our system through web browser. During the evaluation, the code will be compiled into an executable program at the first time. Once the executable is generated, the Executer executes it and check its correctness by comparing with criteria input/output. After verifying the correctness, the assignment is sent to the SonarQube functions to detect its code quality and generates a final score based on both its correctness and quality.

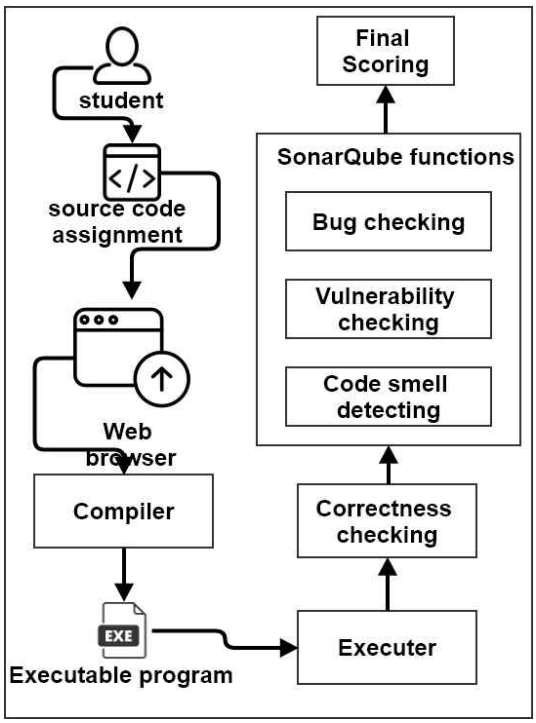

Figure 1. Dataflow of our proposed work

We take advantage of predefined rules in SonarQube and sonar-cxx to detect the common bugs and vulnerabilities. Meanwhile, we also analyzed the submitted C/C++ assignments in the past and synthesize the mistakes students made most frequently and design several custom rules based on them. A few of custom and predefined rules are demonstrated in Table 1 and 2, respectively. To grade the quality more precisely, we give a weight for each rule.

We execute twenty assignment sets on our proposed work which contain 549 assignments submitted by the undergraduate students in the last year. Table 3 shows the most frequent mistakes in the experiment result.

Table 3. three most frequent mistakes

Failed rules	Frequency	Percentage
Missing curly brace	63	72%
Nested statements	18	20%
Collapsible if statement	7	8%
Total	88	100%

4. Discussion

The experiment explains the missing curly braces is the mistake occurred most frequently in the students' submitted assignments. Since such mistakes usually are made not in purpose, it may be appropriate if we point the mistakes by cutting the scores to encourage the students working on improving their code qualities. However, since this paper only provides a prototype of our work, the hypothesis needs to justified further whether such an approach can help students improving their coding skills or not.

Table 4. illustrates the comparison of the C/C++ rules presented in this paper with the total weights of both Java and Python rules presented in our previous work. The differences of the total weights among the three programming languages may lose the consistency since the quality is always important no matter what kind of programming language it used.

Table 4. Comparison of the rules among languages

Language	number of rules	total weights
Java	10	24
Python	4	14
C/C++	9	27

To compare the quality of the codes written in different languages, normalizing the quality scores is necessary. For instance, if we want to compare the quality scores of three different implementations of the same quicksort algorithm, the current quality scores cannot be compared since they are not normalized. Once they are normalized to the same scale, then the relative weights of the similar rules can also be compared across the languages. Even the weights of the rules can be fine-tuned based on this comparison.

Furthermore, some safe coding rules, say MISRA-C or the Secure Coding Guidelines for Java, can be reflected in the quality rules of the coding. For example, one of the rules in MISRA-C suggests the identifiers modified within the increment expression of a loop header shall not be modified inside the block controlled by that loop header. Such rule can also be used for detecting the quality of Java and Python programming. The opposite is also feasible if we'd like to use Java and Python rules in the quality detection for C/C++ programming.

5. Conclusion

We apply SonarQube and its plugin to detect the code quality for the C/C++ programming assignments. It is more accurate to evaluate a programming assignment based on both its correctness and code quality. We evaluate 549 C programming assignments and find out the missing curly brace is the mistake made by students most frequently.

In the future, we are planning executing more submitted assignments from past to analyze the frequent mistakes and find suggestions for instructors to help students avoid such mistakes. In addition, we will keep updating the rules used in the code quality detection to make them consistent for all programming languages.

Acknowledgement

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2014-3-00035, Research on High Performance and Scalable Manycore Operating System).
*Corresponding author: Gyun Woo (Pusan National University, woogyun@pusan.ac.kr).

References

[1] H. Keuning, B. Heeren, and J. Jeuring, "Code quality issues in student programs", in *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, ACM, 2017.

[2] X. Liu, G. Woo, "Using SonarQube to Enhance the Accuracy of Programming Assignments", in *Proceedings of 13th International Technology, Education and Development Conference*, IATED, 2019.

[3] G. Campbell, P.P. Papapetrou. *SonarQube in action*. Manning Publications Co., 2013.

[4] X. Liu, Y. Kim, H. Cho, and G. Woo. "NeoESPA-A New Evaluation System for Programming Assignments". in Proceedings of the 14th International Conference on Electronics, Information, and Communication, IEEE, 2015.

[5] I. Tollin, F. A. Fontana, M. Zanoni, and R. Roveda, "Change Prediction through Coding Rules Violations". in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, ACM, 2017.