An Integrated Automatic-Grading and Quality Measure for Assessing Programming Assignment

Fradina Kristina Sinambela\textsuperscript{1}, Mario E. S. Simaremare\textsuperscript{2}, Evi Fanny Dear Sidabutar\textsuperscript{3}, Rommel Parasian Gultom\textsuperscript{4}

Fakultas Informatika dan Teknik Elektro, Institut Teknologi Del, Laguboti, 22381, Indonesia
e-mail: \textsuperscript{1}iss17067@students.del.ac.id, \textsuperscript{2}mario@del.ac.id, \textsuperscript{3}iss17045@students.del.ac.id, \textsuperscript{4}iss17053@students.del.ac.id

Abstract

Programming teachers very often assess their students’ submissions solely from their correctness, sometimes with the help of an automated-grading platform. This approach is arguably effective and efficient at the same time. Unfortunately, this single-dimension assessment approach narrows the students' perspective on a high-quality solution to just a working solution. In the real world setting, a solution should be maintainable with minimal cost. Maintainability is closely related to code quality. In this paper, we present a web-based tool that would help the teachers to assess the students' submissions from both correctness and quality in one place. In this tool, we integrate GitHub Classroom and SonarQube.

Keywords: automated-grading, code quality, GitHub Classroom, SonarQube

1. INTRODUCTION

A good learning process should end up with a satisfying output. This sort of approach has been embraced and works fine. Unfortunately, assessing the learning process is taking a lot of effort. In programming, for example, the students’ submissions are evaluated from their correctness level; sometimes, manual inspections were conducted to see the internals (the process). Checking the program output is easy; the teacher could design a set of test suites and do the tests or with the help of automated-grading tools. However, checking the internal is a different story. The teacher (with the help of teaching assistants) needs to evaluate the submitted codes and apply the prepared marking scheme or rubrics, not a fun process. This opens to a dangerous domino effect, and the most obvious one is too much time spent in the assessment process. Hence, checking the internals was skipped, and the students are driven by correct-only output. It ends up with a working solution without a known quality level.

Today's learning paradigms are going towards the outcome, not stopping in the output. The result is the effect after the production. In the context of software development, a product could be in the form of its maintainability level, security proneness, speed, etc. Outcome-Based Education (OBE) is an enlightening teaching and learning strategy based upon a predefined set of expected outcomes. OBE may include a range of knowledge, skills, and attitudes. Teaching components and activities should be well organized, planned, and continuously improved [1][2].

Implementing OBE in programming-related courses demands the teachers to do beyond the ordinary. In helping the programming teachers, we argue that equipping the existing automatic-grading tools with more capability is one possible solution. The automatic-grading must cover at least two assessment approaches the solution correctness and the code quality. The code quality directly impacts the software's maintainability level, a form of the outcome. There is an automatic-grading tool called NeoESPA [3] with the ability we needed, but it is not freely available. A very similar tool exists with the same limitation [4] [5].
In this paper, we proposed an alternative option of an automatic-grading tool with the capabilities that we have previously discussed. We integrate two freely available tools (GitHub Classroom and SonarQube) and put them in a tool in the form of an intuitive dashboard.

The rest of this paper is organized as follows. Section 2 describes the research method of the stages carried out in the research. Section 3 illustrates the experiment of using our proposed system. Section 4 discusses the meanings of the experimental results and their limitations. Section 5 concludes our work. Section 6 Acknowledgements.

2. RESEARCH METHODS

2.1. Correctness

In our work, we use GitHub-based automatic-grading in the form of GitHub Action. This grading is also bundled in the GitHub Classroom (GHC) platform. GitHub is a top-rated git-based version control service. It supports collaborative development and project hosting (C. Hsing & V. Gennarell, February 2019). GitHub Action is an event-based automation feature that automates software development workflow; for example, it could be configured to run some tests after a commit occurs and deploy the project somewhere.

In our context, GitHub Action is configured in such a way to run the given test suite on the students' submission repositories. GitHub provides this kind of workflow in the GHC. Through the platform, the teachers can set up assignments. An invitation link is generated for every project. This link is then distributed to the students. To do the work, the students must first accept the invitation link. Once the invitation is accepted, the platform will create a submission repository for the student. Every student will have their store where they then could submit the solution. An assignment can be set as a group assignment. Hence more than one student is enrolled in the generated repository.

When a solution is committed and pushed back to the origin server, the corresponding GitHub Action is triggered. The Action will run the test suite prepared by the teacher. The testing is done in the black-box approach. The final grade is calculated on the points gathered from the test results.

A classroom is simply an instance of a course in a specific period. If a class is available both in the fall and spring, two different classrooms should be created. A classroom is associated with a GitHub Organization. Repositories generated by GHC will be under the associated Organization, and both the Organization's owners and members have access to the warehouses. An organization could be linked to multiple classrooms. The Organization's owners manage a classroom. From the GHC perspective, these managers are called admins[6].

2.2. Code Quality

Code review is a widely used practice to improve the quality of a solution by finding potential defects that were not realized during the development process [7]. These defects may be in dead codes, unsafe resource handling, inefficient algorithms, improper usage of data structures, etc. The more flaws, the lower the quality. Low code quality can cause catastrophic disasters in the long run. It may increase the maintenance cost, decrease the software performance, and ultimately lower the Organization's productivity. The term code smells describe problems related to algorithms and structures that comprise code quality.

Studies suggest guidelines for the students to improve their code and produce a high-quality solution [8] [9]. The guidelines consist of nine distinct criteria spread into four categories described in Table 1.
Table 1. Quality criteria

<table>
<thead>
<tr>
<th>Documentation</th>
<th>Presentation</th>
<th>Algorithms</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Names</td>
<td>Layout</td>
<td>Flow</td>
<td>Decomposition</td>
</tr>
<tr>
<td>Headers</td>
<td>Formatting</td>
<td>Expressions</td>
<td>Modularization</td>
</tr>
<tr>
<td>Comments</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In our work, we employ SonarQube (SQ) to measure the quality of the students' submissions. SQ is a platform that provides numerous functions to measure and manage software quality. It consists of two parts, namely the server and scanner. The SQ server provides an environment for managing and executing data and functions. The platform detects bugs, code smells, and security vulnerabilities for more than 20 programming languages. Besides that, it collects and aggregates all data, such as code coverage, code complexity, unit tests, bugs, and duplications. The platform also provides a dashboard for each project that summarizes the project's quality metrics. It is freely available for selected programming languages [7].

2.3. GitHub Integration Point

GitHub provides a vast number of APIs on top of the HTTP protocols in the form of both REST and GraphQL. Some of these APIs are openly available with rate limitations. Since GHC runs on top of the GitHub platform, we can use the APIs to interact with any resources generated by GHC, including repository, actions, Action runs, etc. By understanding this, we are going to use the APIs as the point of integration.

To access a private or restricted resource in GitHub, authorization is required, which means a legit entity or a user can only issue a request. There are two possible ways to authorize a request; the first option is using the entity's or user's credential (username and password), and the second one by using a Personal Access Token (PAT). The second option is much more preferred because it can be easily revoked when compromised. A PAT is simply a token representing the actual entity or user. In other words, a PAT is the entity's or user's credential but in a form other than username and password. Token-based authentication is commonly used for machine-to-machine interaction rather than human-to-machine.

To enable this approach, we need to provide a GitHub account to represent our machine. This account is then added as a member to the Organization where the classroom is bound to. Afterward, we generate a PAT representing the account with read-only access. This PAT then can be used to authorize any API request from our machine.

Based on our investigation, we found that when a student submits their work (via his/her repository), the automatic-grading Action associated with the repository is executed, and the result or output is stored as a Check Run. The repository, the Action, and the Check Runs can be seen as interrelated resources. By understanding these relationships, we can focus on a handful of API endpoints related to these resources. Information related to the grading can be retrieved through REST APIs provided by GitHub (Table 2).

Table 2. API endpoints related to GitHub Classroom

<table>
<thead>
<tr>
<th>No.</th>
<th>REST API Pattern</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>GET /orgs/{org}/repos</td>
<td>Repos</td>
</tr>
<tr>
<td>2.</td>
<td>GET /repos/{owner}/{repo}/actions/runs</td>
<td>Actions</td>
</tr>
<tr>
<td>3.</td>
<td>GET /repos/{owner}/{repo}/checksuites/{check_suite_id}/check-runs</td>
<td>Checks</td>
</tr>
</tbody>
</table>
Fradina, et., al

An Integrated Automatic-Grading and Quality Measure for Assessing Programming Assignment

API description:

- The GET /orgs/{org}/repos endpoint
  This endpoint is used to obtain data related to the repository.

- The GET /repos/{owner}/{repo}/actions/runs endpoint
  This endpoint is used to obtain data on the actions associated with a repository.

- The GET /repos/{owner}/{repo}/checksuites/{check_suite_id}/check-runs endpoint
  This API is used to obtain the result of the executed automatic-grading Action.

2.4. SonarQube Integration Point

Sonar Qube (SQ) provide Sonar APIs. Sonar uses APIs to manage account access permissions so that the system can run programs. SQ can be linked with an existing workflow to allow continuous code inspection across all project branches and pull requests. To access a project in SQ, Token as a replacement for the user login is required for permission. By not allowing the password of the analysis used to flow via the network, the installation will be more secure. To make it easier for the user to provide the token name, the system will produce a token that SQ can use [10].

SonarQube presents data quality code execution results at the student program source code via web APIs on the webserver, and then the data is obtained in JSON format. With this understanding, The APIs will be used as a point of integration (Table 3).

<table>
<thead>
<tr>
<th>REST API Pattern</th>
<th>Endpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET /orgs/{org}/repos</td>
<td>Repos</td>
</tr>
</tbody>
</table>

Table 3. SonarQube API pattern to get the results

3. DESIGN

All of the GitHub repositories can be cloned/downloaded, and stored on a local computer. Cloning is done by using a token configured on the local computer to call a private repository. The cloned repository will be placed in a local computer directory which will then be used during the code quality aspect assessment process. Data storage on a computer is done by creating a folder that matches the assignment name of the task, then the repository associated with the job will be stored in it. The following of the storage structure of the clone results in the directory (Figure 1).

Figure 1. Storage Structure

Figure 2 depicts the system architecture exposing how the integrations are tailored between tools. As been discussed in the previous chapter, the integrations are possible through REST APIs both GitHub and SonarQube.
4. RESULT AND DISCUSSION

4.1. Result of Correctness Aspect

Figure 3 shows the web interface of the integrated tools. The system is developed on top of the Python and powered with a SQLite database. The system accepts two inputs, an organization name associated with a classroom, and the assignment name.

Under the hood, the system will issue a sequence of requests to GitHub asking for the repository information, the executed Actions, and the grades. Afterwards, the system will clone the repositories (Figure 4) onto the local storage with structure described in Figure 1.
Once the repositories are cloned, the system triggers SQ to run the code quality assessment. When the assessment is finished, the last thing to do is retrieving the results and join them with those retrieved from GitHub. The joined results are then shown on the web page.

4.2. Discussion

In this section, based on the results of the Correctness aspect, the system has grabbed the GitHub Classroom auto-grading value via PATs and provided the name of the repository that matches the search results in the search field. After getting the GitHub Classroom auto-grading value, the system has also succeeded in saving the repository data on the user's local computer, so that the repository will be stored based on the assignment name for use in assessing the code quality aspect.

Checking the code quality aspect is carried out directly through Personal Tokens and sending assessment metrics through the Sonar API. However, the assessment of the code quality aspect through SonarQube cannot be carried out properly if the program code is empty, so this has been overcome by deleting the empty program code program file in the implementation.

5. CONCLUSION

This paper presents an alternative tool to help the teachers to grade their students’ programming submissions. Our system can grade the submissions from two angles: the correctness and quality aspects. We combine GitHub Classroom, a freely available automatic-
grading tool, and SonarQube to measure the code quality. This tool will be made available for everyone at no cost. In the future, we would like to integrate other assessment aspects for example academic integrity, effort-based measurements, etc. It would also be great if more

6. ACKNOWLEDGEMENTS

This research and publication were supported and fully funded by LPPM Institut Teknologi Del with contract number 045.15/ITDel/LPPM/Penelitian/IV/2021.

REFERENCES


