Improving Programming Education Quality with Automatic Grading System

Yun-Zhan Cai(✉) and Meng-Hsun Tsai

Department of Computer Science and Information Engineering,
National Cheng Kung University, Tainan, Taiwan
caiyz@imslab.org

Abstract. As the rapid growth of information technology, the demand for proficiency in software programming skyrockets. Compared to teaching with slides traditionally, hands-on programming training is more beneficial and practical. However, it is exhausting and time-consuming for educators to grade all assignments in person. Besides, students may not get feedback immediately to correct their wrong conceptions. Therefore, an automatic grading system is required to grade and send feedback to students. Based on an existing continuous integration system, which checks whether new programs behave as expected, we develop a set of course management tools and deploy an automatic grading system in this paper. Our system requires a server to run and test the programs. However, the server is susceptible to being compromised by hackers. Therefore, how we protect sensitive data and prevent malicious network traffic are demonstrated in this paper as well. The tools were applied in an Android application development course with 140 students enrolled. Around 72% of the students indicate the automatic grading system is beneficial to their learning.

Keywords: Android · Continuous integration · Programming education · Software design

1 Introduction

As the rapid growth of information technology, the demand for proficiency in software programming skyrockets. Compared to teaching with slides traditionally, hands-on programming training is more beneficial and practical. For example, students are able to get acquainted with development tools and consider problems more deeply during the course of hands-on programming. Besides, as the saying goes, “practice makes perfect”. Students can gain precious experience after troubleshooting problems. Unfortunately, the number of students is much more than that of educators in most cases, which in turn makes educators be overwhelmed with the heavy workload. Especially for an Android application development course, each build takes a few minutes. Building all Android projects to grade students’ assignments is extremely time-consuming. Students cannot obtain feedback in a short time if their assignments are not reviewed immediately. What’s worse, conceptions in the class are coherent mostly. If educators are not able to correct students’ wrong conceptions, it is likely that students will struggle to keep up with the class. Therefore, there are still lots of room

https://doi.org/10.1007/978-3-030-35343-8_22

for improvement on training approaches of hands-on programming, for instance, how to effectively track learning status of students and give immediate feedback to them.

In order to track learning status of students, there are some cases that the educators apply version control tools such as Git [4] to the hands-on programming courses, and ask the students to upload their source codes to the source-code-hosting facility such as GitHub or SourceForge [3, 5, 10]. This approach not only helps the educators manage students' assignments more easily but also help the students familiarize conceptions of the version control and share their projects with others. Nowadays, GitHub has launched ClassRoom service, which allows educators to establish associations between students and their GitHub accounts [8]. Besides, ClassRoom provides a convenient way to download all projects in single click as well. However, we found the following problems during a trial of ClassRoom. First, importing the starter codes sometimes fails, which was reported and discussed before. This situation implies that GitHub ClassRoom is still unstable. Second, ClassRoom imports all branches from the starter repository, which forces educators to place questions and answers of an assignment in separate repositories instead of in separate branches. Otherwise, Students are able to view the answer branch, which is imported along with the question branch. Storing questions and answers in different repositories may also produce redundant and messy codes, which are not easy to organize. Last, after educators update some files in the starter repository, the modifications are not synchronized to students' repositories. This limitation hinders educators in adding or modifying questions, which means that ClassRoom is not flexible enough to meet the update requirements. Given the three problems mentioned above, how to improve programming hands-on courses requires more studies.

Continuous integration plays a significant role for enterprises and open source software communities of today [6, 12, 13]. After a software developer pushing a new commit to the source-code-hosting facility, the continuous integration system will build and check if there is any problem automatically, which assists the developer to find out bugs as soon as possible. Hence if we can apply the continuous integration to the hands-on programming courses, the students will be able to realize and correct their mistakes in the assignments according to the error messages in the build logs. There are plenty of continuous integration platforms nowadays such as Jenkins, Travis, Circle CI and so on [20]. Among them, Travis has built the GitHub App and sold it in the GitHub Market. That makes the workflow automation process of Travis easier. Furthermore, Travis supports the Google App Engine deployment, which allows educators to get rid of difficulties in setting up a server to run continuous integration system [11]. After applying for an educator discount, educators are able to create private repositories under the organization and use Travis for free [19]. On the basis of the reasons we mentioned above, we think Travis is highly suitable to serve as the grading platform for programming courses.

Since the technique for applying continuous integration to the software hands-on programming is still incomplete, we develop a set of course management tools, consisting of `CourseManager.py`, `result_collector.py` and `grade.py`, based on an existing continuous integration system to make assignment management easier. (The `.py` extension means the tools are written in Python language which is supported by most operating systems nowadays.) Besides, educators need to connect servers or devices outside Travis sometimes. For example, we speed up builds in Travis by using an external Android emulator instead of creating a new one in Travis for each build. However, it is dangerous to expose a server to the Internet without any defense mechanism. The server is susceptible to be attacked [7]. Therefore, we will demonstrate an example to protect sensitive data and block malicious network traffic in this paper.

The remainder of the paper is organized as follows. In Sect. 2, we take an Android course for example and give an overview of our system architecture. In Sect. 3, details of the system are discussed along with figures. In terms of system stability, security issues concerning the system are described in Sect. 4. In Sect. 5, opinions of students who used our system in an Android application development course are showed. Finally, Sect. 6 concludes our paper and lists some future works.

2 System Architecture

2.1 Use Case of an Android Course

Figure 1 illustrates a use case diagram of an Android course. First of all, a student pushes a new commit to GitHub. GitHub then send a notification to the server though the webhook along with information including a list of modified files, commit hash, commit message, etc. [9]. After receiving the notification, the server checks if the student modifies some sensitive files such as the `.travis.yml` and update the cheating records to the database. At the same time, Travis pulls the new commit from GitHub and start to build according to the settings defined in `.travis.yml`. Eventually, Travis sends the result of the build to the server, and then the result is saved in a Sqlite database. Therefore, educators are able to track status of students' assignments whenever they need.

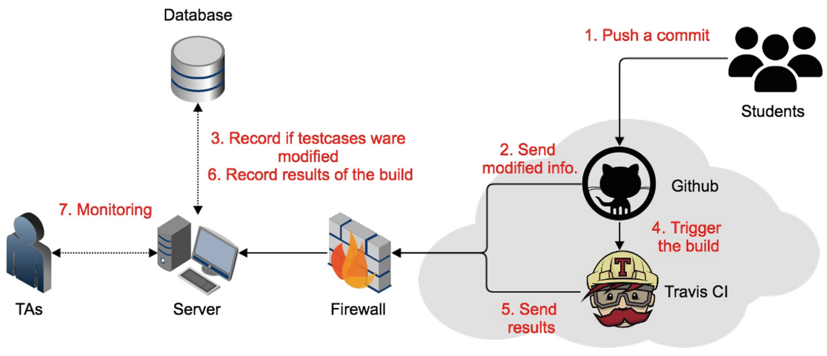


Fig. 1. Use case diagram of an Android course

2.2 Workflow

Figure 2 shows the workflow of the assignment management. First, educators create a repository for the starter codes and add testcases, which are the programs used to check whether students' programs correct or not, to the repository. Afterwards, educators edit `.travis.yml` and `testcases.yml` so that Travis and the server can build the project and record the result respectively. With the CourseManager.py, educators can execute a series of commands such as creating students' repositories, importing starter codes (copy all files from the default branch of the starter repository to the students' repositories) and inviting students as collaborators. If there is any error in testcases, educators can push new commits to correct the error with CourseManager.py as well, and the modifications will be synchronized to the students' repositories. In the end of school term, scores of each assignment can be exported from the database with another tool named `grade.py`, which prevents educators from grading all assignments manually.

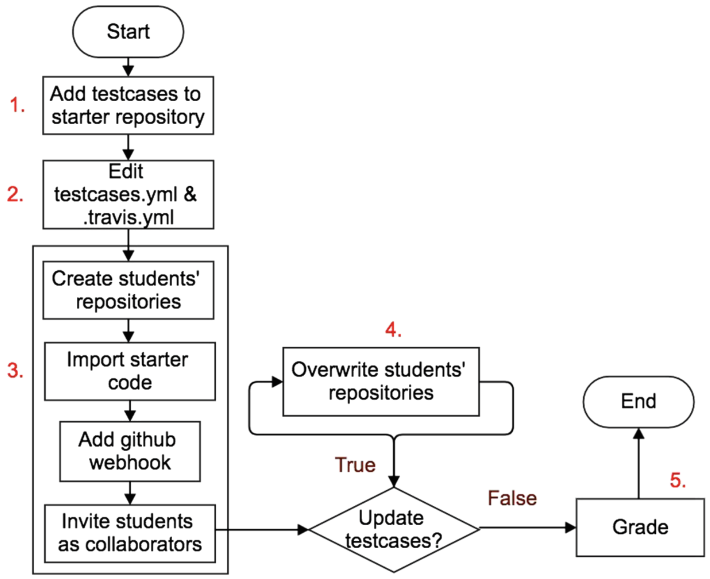

Start
1.
Add testcases to starter repository
2.
Edit testcases.yml & .travis.yml
3.
Create students' repositories
Import starter code
Add github webhook
Invite students as collaborators
4.
Overwrite students' repositories
True
Update testcases?
False
End
5.
Grade

Fig. 2. Workflow of the assignment management

3 Details in Workflow of Assignment Management

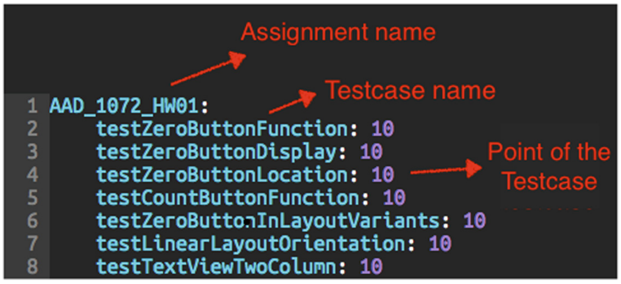
In this section, we describe the details of each step in Fig. 2 in sequence.

3.1 Add Testcases to the Starter Repository

Testcases are the programs which are used to check if there is any logic error in students' assignments. We can classify testcases into two categories: unit test and user interface (UI) test. The former makes sure a section in the program generates expected outputs according to corresponding input arguments. For example, We can program a unit test, `test_add()`, to verify whether the function, `add(a, b)`, returns the summation of `a` and `b` correctly. The latter ensures components in the screen react correctly when a user interacts with them. For example, after a user clicking a button on the screen, an app should be launched. Testing libraries vary from a programming language to another. Take Android for instance, testing libraries for the unit test include Junit [16] and Robolectric [15]. The former aim to test the framework of java and logic of functions. The latter aim to emulate Android components on the JVM and check the configurations of the components. Testing libraries for the UI test in Android include Espresso [1] ad UI Automator [2]. The former is only used inside an application. If there is a need to test components outside an application such as notifications, then the latter should be used instead of the former. In this step, educators have to translate the grading criteria to the testcases and upload the testcases to the starter repository, and then the students' assignments will be tested as specified by testcases.

3.2 Edit `.travis.yml` and `testcases.yml` Files

In order to let Travis build and test automatically and save the results sent from Travis to the database, educators have to edit two files: `.travis.yml` and `testcases.yml`. After preparation of both files has done, educators are able to start the server called `result_collector.py`.



```
AAD_1072_HW01:
    testZeroButtonFunction: 10
    testZeroButtonDisplay: 10
    testZeroButtonLocation: 10
    testCountButtonFunction: 10
    testZeroButtonInLayoutVariants: 10
    testLinearLayoutOrientation: 10
    testTextViewTwoColumn: 10
```

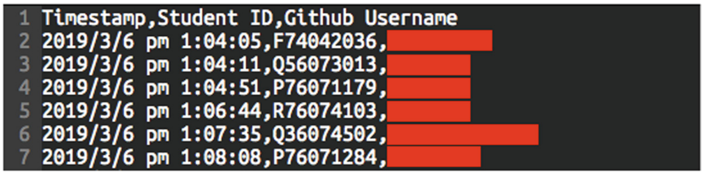
Fig. 3. Example of the testcases.yml

The `.travis.yml` is a configuration file of Travis, which defines the executed instructions to build the project. After a programmer pushes a new commit to GitHub with the `.travis.yml`, the build will be triggered immediately. A job in Travis contains two parts: install and script [18]. The installation of Android SDK is in install part, and an emulator is connected at the part of `before_script`. With respect to the part of `script`, `gradlew` is used to build the Android project. As to the part of `after_script`, we send the result of the building to the server by HTTP POST.

As for the `testcases.yml`, it records the information regarding the testcases of the assignments and is placed under the same directory as the server programs. The `testcases.yml` defines information which is used to create and interact with Sqlite database. Without the knowledge regarding database, educators only need to fill out `testcases.yml` with the assignment names, testcase names and the point of each testcase as shown in Fig. 3, and the server will create the database table that contains results of tests, commit time, branch and commit hash, etc.

3.3 Create Students' Repositories

In this step, educators place the `students.csv` under the directory same as the server programs and utilize CourseManager.py to do a series actions to students' repositories.



```
Timestamp,Student ID,Github Username
2019/3/6 pm 1:04:05,F74042036,[redacted]
2019/3/6 pm 1:04:11,Q56073013,[redacted]
2019/3/6 pm 1:04:51,P76071179,[redacted]
2019/3/6 pm 1:06:44,R76074103,[redacted]
2019/3/6 pm 1:07:35,Q36074502,[redacted]
2019/3/6 pm 1:08:08,P76071284,[redacted]
```

Fig. 4. Example of students' IDs and GitHub accounts

CourseManager.py is a command-line interface tool written in Python. It communicates with GitHub and Travis after authenticated by tokens. Educators can create students' assignment projects, import starter code and add webhook with `start` command, which requires two positional parameters: the name of assignment and starter repository. Considering creating students' assignment projects consumes lots of time, we separate `start` and `invite` commands and execute them independently. By doing so, educators can prepare all assignment projects in advance before class and invite all students as collaborators when announce the assignment.

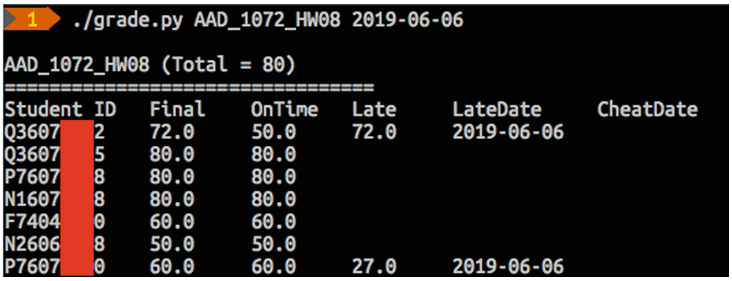
As shown in Fig. 4, `students.csv` contains students' IDs and their GitHub accounts, which is collected through Google form, in order to create students' repositories and invite the students as the collaborators. When educators create the students' repositories with CourseManager.py, CourseManager.py reads relation between students' IDs and GitHub accounts of the students from the `students.csv`, and then create the students' repositories.

3.4 Update Testcases

If educators want to modify some files in the starter repository, all educators need to do is to push a new commit to the starter repository and execute the `overwrite` command. Similar to `start` command, the `overwrite` command has two positional parameters as well. Modified files including testcases, `.travis.yml`, `README.md` will be synchronized to students' assignment projects.

3.5 Grade

Figure 5 is an example of using the grading tool. After specifying a name and deadline of an assignment, educators can export scores of the assignment via `grade.py` tool, which calculates the scores of the students by reading from the database and accumulating the point of each testcase. Later, `grade.py` will adjust the scores of the students according to the status of the late work and the regulation. Take our Android course for example, if a student's assignment is delayed for **n** weeks, the score of the student will be 0.9^n of the original score. We choose the larger one between the original score and the late score as the final score. If there is a student who modify sensitive files such as testcases and `.travis.yml`, which results in unfair grading, then the date when the student modify sensitive files will be shown in the output.



```
 1  ./grade.py AAD_1072_HW08 2019-06-06

AAD_1072_HW08 (Total = 80)
=================================
Student ID   Final    OnTime   Late     LateDate     CheatDate
Q3607[redacted]2     72.0     50.0     72.0     2019-06-06
Q3607[redacted]5     80.0     80.0
P7607[redacted]8     80.0     80.0
N1607[redacted]8     80.0     80.0
F7404[redacted]0     60.0     60.0
N2606[redacted]8     50.0     50.0
P7607[redacted]0     60.0     60.0     27.0     2019-06-06
```

Fig. 5. Example of grading

4 Security Issues

4.1 Encrypt Environment Variables

To prevent attacks from the Internet, sensitive data such as the server's IP or URL, which results of build are sent to, cannot be revealed in `.travis.yml`. The educators can edit the dictionary, `TRAVIS_SECURE_VAR`, in the CourseManager.py, and then the keys and values in `TRAVIS_SECURE_VAR` will be encrypted before added to the `.travis.yml` [17]. In this way, only Travis is able to decrypt and use these secure variables.

4.2 Configurations of the Firewall

To speed up builds in Travis, we run an Android emulator on the server and instruct Travis to connect our emulator before building. While using the remote Android emulator shortens the execution time of a build indeed, the emulator with exposed ports is likely be scanned by a botnet and then compromised. Hence, we execute the commands in Table 1 on the server, which inserts some rules to the iptables of the server and filters out the network traffic from the devices other than Travis to the emulator [14].

Table 1. Commands of iptables

Step	Command
1	sudo iptables -A INPUT -s localhost -p tcp –dport <port> -j ACCEPT
2	sudo iptables -A INPUT -s nat.travisci.net -p tcp –dport <port> -j ACCEPT
3	sudo iptables -A INPUT -p tcp –dport <port> -j DROP

5 Result of Applying Automatic Grading System on Programming Education

Our system was applied in an Android application development course with 140 students enrolled. At the end of the course, we sampled about one third of the students and asked for their opinions on the automatic grading system. The statistical result shows that 72% of the students speak highly of the system. Most of the students tell that an immediate feedback helps them to figure out whether there is any mistake in their programs. Unfortunately, some students reflect the grading logic is not flexible enough, which limit implementation methods of some functions. The extent of limitations depends on how an educator translates grading criteria to testcases. For example, if the goal of an assignment is to create a square, however a student creates a rectangle instead. The student cannot get any point for the testcase just because of the wrong shape. To solve the problem, the educator should claim the goal of assignments clearly before students start doing their assignments. Generally, we think our system is able to improve programming education quality.

6 Conclusion and Future Works

Hands-on programming training is indispensable to cultivate a skillful software developer. However, a system for hands-on programming training is still immature. We aim at designing a system which can help educators manage assignments more easily and provide a better learning experience to students.

In this paper, we develop a set of course management tools based on an existing continuous integration system. Compared with GitHub ClassRoom, our course management tools enable educators to update and synchronize files in the starter repository. Besides, our tools allow educators to place the question and answer branches in the same repository. With our course management tools, educators are able to manage assignments more flexible and stable. A better learning experience can be provided to students as well. Students are able to find errors in their assignments and correct wrong conception immediately. In terms of security issues, we encrypt sensitive environment variables before adding them to the `.travis.yml`. Furthermore, the firewall is set to filter out the malicious network traffic, which prevents our system from being attacked by hackers. After using our system in an Android application development course, 72% of students indicate that the automatic grading system is beneficial to their learning.

In the future, we plan to create a website with Javascript libraries such as D3.js or JQuery for educators. Compared to the CLI, operations on the website are more intuitive for educators. Instead of presenting the information of the students with plain texts, we are going to show the information with charts. For example, the distribution of students' scores can be plotted with a histogram, and the error rate of each testcase can be plotted with a pie chart. In this way, educators are able to track the status of the students faster than before. What's more, after analyzing the status of a student with the data mining technology, we can recommend some additional materials to the student.

After enhancing the user interface and the data analysis, we believe we can build a sounder learning platform for educators to provide a better learning experience to students.

Acknowledgement. This work was supported by the Center for Open Intelligent Connectivity through the Featured Areas Research Center Program within the Framework of the Higher Education Sprout Project by the Ministry of Education in Taiwan. The work of Y.-Z. Cai was sponsored by the R&D enhancement project "R&D of Network Behavior Security Analyses for IoT Devices on Advanced Edge Switch in an AIOT plus SDN Integrated Platform," which is executed by EstiNet Technologies Inc. and partially sponsored by Hsinchu Science Park Bureau, Ministry of Science and Technology, Taiwan, R.O.C. The work of M.-H. Tsai was supported in part by the MOST under Grant 107-2221-E-006-062 and Grant 108-2221-E-006-112, and in part by the Industrial Technology Research Institute.

References

1. Android_Developers: Espresso. https://developer.android.com/training/testing/espresso
2. Android_Developers: Ui automator. https://developer.android.com/training/testing/uiautomator
3. Britton, J., Berglund, T.: Using version control in the classroom. In: Proceeding of the 44th ACM Technical Symposium on Computer Science Education, pp. 753–753. ACM (2013)
4. Chacon, S., Straub, B.: Pro GIT. Apress, New York (2014)
5. Clifton, C., Kaczmarczyk, L.C., Mrozek, M.: Subverting the fundamentals sequence: using version control to enhance course management. ACM SIGCSE Bull. **39**(1), 86–90 (2007)
6. Cusumano, M.A.: Extreme programming compared with microsoft-style iterative development. Commun. ACM **50**(10), 15–18 (2007)
7. Felt, A.P., Finifter, M., Chin, E., Hanna, S., Wagner, D.: A survey of mobile malware in the wild. In: Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, pp. 3–14. ACM (2011)
8. GitHub_Inc.: Github classroom. https://github.com/education/classroom
9. GitHub_Inc.: Github developer: webhooks. https://developer.github.com/webhooks/
10. Glassy, L.: Using version control to observe student software development processes. J. Comput. Sci. Coll. **21**(3), 99–106 (2006)
11. Google: Implementing continuous delivery with Travis CI and App Engine (2019). https://cloud.google.com/solutions/continuous-delivery-with-travis-ci
12. Holck, J., Jørgensen, N.: Continuous integration and quality assurance: a case study of two open source projects. Australas. J. Inf. Syst. **11**(1), 45 (2003)
13. Miller, A.: A hundred days of continuous integration. In: Agile 2008 Conference, pp. 289–293. IEEE (2008)
14. Rash, M.: Linux Firewalls: Attack Detection and Response with iptables, psad, and fwsnort. No Starch Press, San Francisco (2007)
15. Robolectric: Robolectric, http://robolectric.org/
16. The_JUnit_Team: Junit (2019). https://junit.org/
17. Travis: Encryption keys. https://docs.travis-ci.com/user/encryption-keys/
18. Travis: Job lifecycle. https://docs.travis-ci.com/user/job-lifecycle/
19. Travis: Travis CI education. https://education.travis-ci.com/
20. Watters, C., Johnson, P.: Version numbering in single development and test environment (2013)