Title: Using Github Classroom to teach unit testing

Author: Chenhao Li

Degree: MSc Computing

Supervisor: Carl Jones

Institution: School of Computer Science and Informatics, Cardiff University

Date: 2020/09/23

Table of Contents

Abstract

There are many online learning management systems(LMS) widely used by universities to improve teaching efficiency and learning quality. The Github Classroom is a teaching platform launched by Github in 2016 with functions such as version control, automatic assessment, and quick feedback. The article introduces significant features, functions, and workflow of Github Classroom and implemented the system to teach unit testing, TDD, and BDD. A form of Plan-Do-Check-Action(PDCA) loop is used to conduct a series of teaching experiments. Most of the advantages and limitations of Github Classroom are discussed, while some suggestions are provided for both teachers and students to use it. According to the strengths of the intelligent system, I highly recommend the teachers who want to reduce their workload of evaluation and increase students' practice of coding to use Github Classroom.

Key words: Github Classroom, Unit Testing

Acknowledgement

First and foremost, I would like to deeply appreciate my supervisor Mr. Carl Jones for his guidance and encouragement in carrying out this project. He always gives me helpful suggestions, it would be difficult to complete research without his help.

Secondly, I would like to express my thanks to all professors and tutors who taught me in the postgraduate period.

Finally, I am also thankful to my parents and friends who helped me throughout the project.

Introduction

As the demand for higher education grows, and the university classes increase, the teaching workload of university teachers is getting heavier. Online learning management systems (LMS) such as Google Classroom, Canvas, Moodle, Sakai are being widely used in higher education as necessary supplements of traditional face-to-face teaching to lighten teachers' workload and improve teaching quality. Github Classroom is a teaching platform launched by Github in 2016 with functions such as version control, automatic evaluation, and quick feedback. Now a large part of the workload is to evaluate students' assignments and exams and provide timely feedback, but the majority of the assessment tasks are manual (Ala-Mutka 2005). So Github Classroom may possibly become an excellent option for teachers to improve teaching efficiency while helping students to learn better. However, relatively fewer discoveries have been focused on Github Classroom since it is a new system. It is valuable to research Github Classroom to provide information about its features and potentials for universities and teachers who want to learn and use LMS.

The primary goal is to explore the essential functions and how to use Github Classroom, discuss what this platform is good at and what are defects, provide useful suggestions and recommendations for both teachers and students. The study is going to introduce the entire workflow of using Github Classroom, including how to create a classroom or an assignment and how to use automatic assessment and provide feedback, which is considered to be effective methods to improve teaching quality and efficiency. In the second section, a series of teaching experiments are conducted in the form of the Plan-Do-Check-Action(PDCA) loop to demonstrate how to set up a complete course using Github Classroom and explain my own experience of teaching Python unit testing. Finally, the advantages and limitations are discussed to give advice.

Git is a popular tool for version control. Github is an open-source platform for

hosting software projects using Git (Loeliger and McCullough 2012). Github Classroom is an online teaching system launched by Github, which is convenient for managing teaching resources, with automatic evaluation, quick feedback and other functions (Fiksel et al. 2019).

The author has a background in computer science, including programming language such as Python, but does not have any experience of using an online teaching system. After learning the Github Classroom, I believe that this system has many excellent functions and advantages both for students and teachers. With further learning the educational concepts such as formative assessment and summative assessment, I am willing to explore how Github Classroom supports both formative assessment and summative assessment. A summative assessment is an evaluation of the educational quality of a school section or a subject. The purpose is to make a conclusive evaluation of students' learning with a result that to draw conclusions for students or to give grades (Harlen and James 1997). Formative assessment is the evaluation of students' performance and achievements in the daily learning process, as well as the development of emotions, attitudes, and strategies. It is also based on continuous observation, recording, and reflection of the entire learning process (Black and Wiliam 2009).

Unit testing refers to the inspection and verification of testable units, such as methods in the software. Python unittest is a built-in module for unit testing, which is easy to use (Runeson 2006). Test-Driven Development(TDD) is a core practice and technology in agile development, as well as a design methodology. The principle of TDD is to write unit test code before developing functional code and use the tests to drive the design of product code (Beck 2003). Behavior-driven development (BDD) is an extension of TDD, which converts structured natural language sentences into executable tests, has a closer relationship with a given function. As a design method, BDD can effectively improve the design and indicate the direction for the team during the evolution of the system (Soeken et al. 2012). Behave is a mature BDD framework that supports the writing and testing of use cases in natural language. The author used Github Classroom to teach Python unittest, TDD, BDD, and Behave,

tried to discover Github Classroom, and establish a teaching course of unit testing.

In the process of exploring Github Classroom and teaching Python Unit testing, I discovered and thought from the perspectives of teachers and students, tried to provide suggestions and advice for both teachers and students. A teacher might be more concerned about whether the teaching system is convenient and efficient to use, whether it reduces the workload. Students possibly are more focused on whether the platform could support good teaching quality and whether the automatic assessment could help them improve learning efficiency. Teachers and students both attach great importance to the timeliness of feedback, which is the basis for enhancing teaching and learning and the core of communication.

Manual review means a tremendous amount of work. Teachers should not only review the quality of the students' code, but also check whether the code is running normally, and confirm the results of the code. Automatic assessment can complete the evaluation of any assignment in a few seconds, which significantly improves efficiency. But there are still some risks in the automatic evaluation. Reducing manual intervention means that the accuracy and consistency of the assessment results could not be fully guaranteed, and this may increase the possibility of cheating (Ihantola et al. 2010).

Workflow of Github Classroom

I will now provide an introduction to the workflow of Github Classroom from both a teacher and student view in this chapter. Teachers need to set up classrooms and assignments which are easy and convenient to do, and students can start learning tasks as long as they accept the assignment.

1. Creating Classroom

The first step is to create the classroom, select an organization for the classroom and invite collaborators and students.

1.1 Organization

GitHub Classroom uses the organization to manage permissions, administration, and security for classrooms. Managers could invite Github users to their organization or send invitation URL.

Figure 1. Organization

1.2 Roster

The student roster helps to easily track student progress on the dashboard. GitHub Classroom is able to automatically import a roster from an institution or a learning management system. A CSV or text file could be uploaded to create a roster by listing students.

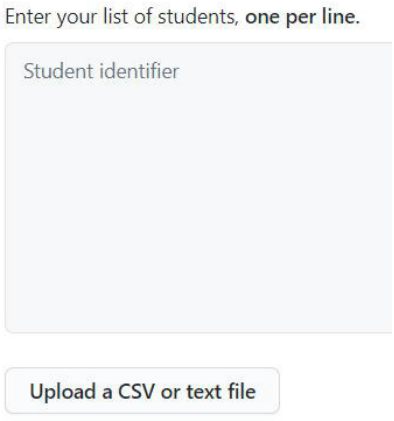

Figure 2. Roster

2. Creating Assignment

The assignment is one of the most important functions of the Github Classroom. Lessons or exercises are distributed to students in the form of assignment through an invitation URL. The title, deadline, type of assignment, visibility of repository could

be edited.

2.1 Starter code

Lecturers could set a template repository to give students a starter code. Pre-written teaching materials could be added into the repository for students to read, which could be in any readable formats such as pdf, txt, md, etc.

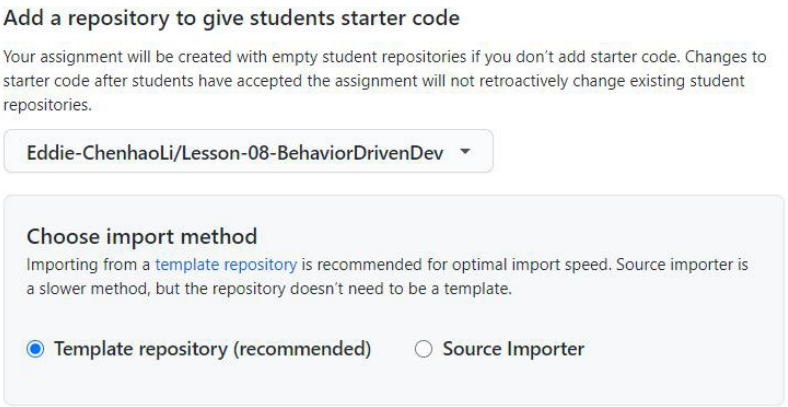

Figure 3. Starter code

2.2 Autograding tests

Autograding tests help provide feedback for students immediately upon submission using GitHub Actions. The grading methods could be an Input/Output test, a command, or a test case of different languages such as Java, Python, C, etc. Each grading method is an individual task, called action, and could create a custom workflow.

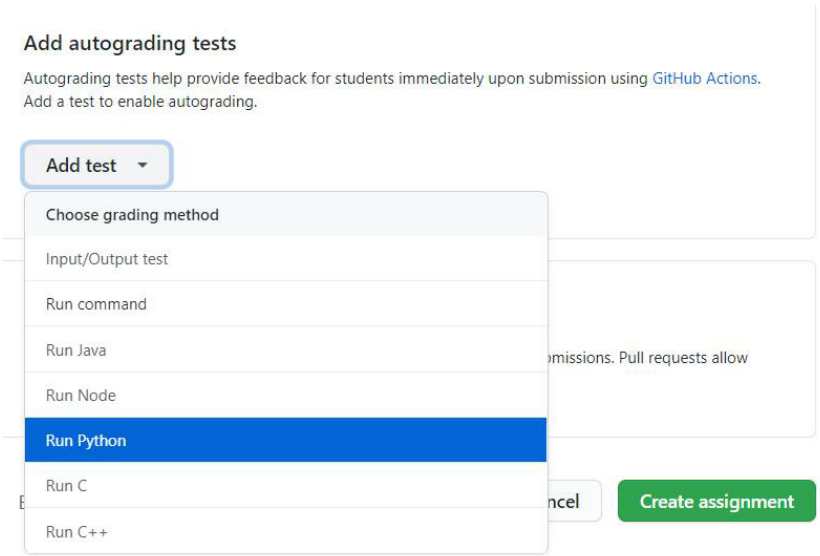

Figure 4. Autograding tests

3. Review and feedback

3.1 Overview of submissions

After students complete the assignment, lecturers could see the overview of submissions on the assignment page. As the Figure 5 illustrated, On the left side of the list, it shows whether the student passed all the tests in the last submission, next to it is the students' score in this automatic assessment. The number of submissions by students is shown on the right.

Figure 5. Overview of submissions

3.2 Workflow of testing

Every commits from students could be seen on the Review page, and the entire test workflow could be seen on the Workflow page. The number of students' submissions and the submission time can reflect some vital information. For example, some students may submit more than ten updates, and each update is wrong, then the student might have encountered some difficulties in the current course. If a student can get full marks on the first submission for every assignment, the student could be particularly good or may have copied the answer. The complete test workflow can be checked both by teachers and students. For students, a test workflow is an excellent tool for debugging. Students can use it to improve their ability to read the error log and the capability of error locating. Code development and error debugging are equally crucial in programming.

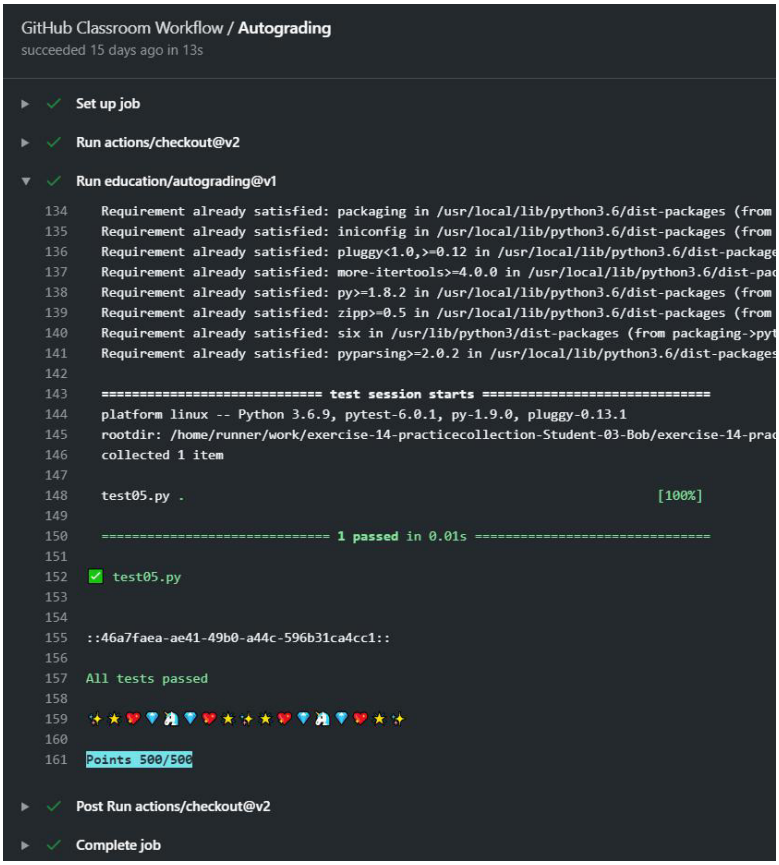

Figure 6. Workflow of testing

3.3 Online feedback

Lecturers could write a report for students on the feedback page or even leave a comment on a certain line of code to give a detailed suggestion.

Figure 7. Leaving a comment

4. Student use

4.1 Accepting assignment

Students will automatically create an assignment repository in Github after receiving

the invitation link and accepting the assignment, which includes the tutorial documentation written by lecturers and the initial code of the project.

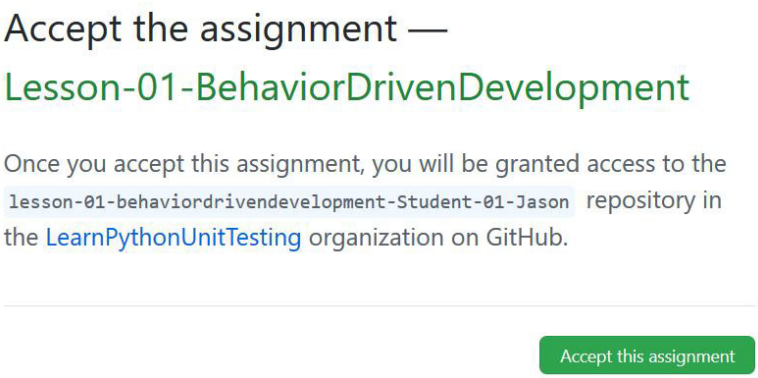

Figure 8. Accepting assignment

4.2 Commit assignment

Students can clone the repository to local for development and submit code, which is a recommended way for students to use it, which can make students more proficient in version control and Git operations. Students can also edit and submit code directly on the web page.

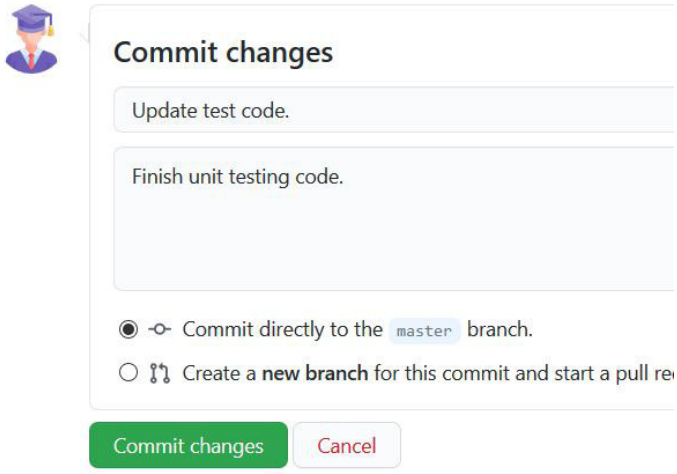

Figure 9. Commit changes

Experience of teaching unit testing

I am now going to describe the work and my own experience of teaching unit testing, Test-Driven Development, and Behavior-Driven Development in this chapter. For conducting a series of teaching experiments, the PDCA cycle would be a suitable

management method. The meaning of the PDCA cycle is to divide quality management into four stages, which are Plan, Do, Check, and Action. In the quality management activities, each work is required to establish objectives, implement plans, check the results of implementation, and finally back to plan if the results are unsatisfactory or standardization if the results are satisfactory (Moen and Norman 2006). I followed the PDCA loop to determine the plan and the goal before each experiment, then create lessons and assignments according to available features and functions of Github Classroom, adjust them based on the changes of design and implement the contents in the plan. In the Result section, I will summarize the results of this construction, analyze the strengths and weaknesses, find out the problems, and compare the results and expectations. Finally, I will make a plan for the next cycle based on the problem, and put the temporarily unsolvable problems into the Still Unknown section.

1. Experiment - Can Github Classroom introduce unit testing?

Goal	To access student ability to write tests for given program code
Teacher provides	Knowledge about unit testing and program code
Students need to	1. Write tests based on program code 2. Make tests pass

1.1 Plan

In this experiment, I was trying to:

i. make students be familiar with the teaching system
ii. make students learn basic knowledge about unit testing

Therefore, as the first course, it needs to provide students with sufficient theoretical knowledge about unit testing, while the exercises could be set to be simple and easy to understand.

1.2 Process

In the first course, detailed tutorial documentation with plenty of classic test examples is given for students to read and understand. I set a series of exercises for

students to learn unit testing from easy to difficult. The difficulty of exercises not only depends on the complexity of coding but also on the knowledge of unit testing, including different assertion statements, inquiry methods, function decorators, class decorators, and Mock library for testing. Students are required to write the tests according to the program code and make the tests pass, which is one of the most basic and efficient ways to practice writing tests. Take one of the lessons as an example (Example URL: https://github.com/Eddie-ChenhaoLi/Lesson-05-Mock):

```python
class Person:
    def __init__(self, first, last, age):
        self.first = first
        self.last = last
        self.age = age

    def get_full_name(self):
        return '{} {}'.format(self.first, self.last)

    def is_adult(self):
        if self.age >= 18:
            return True
        else:
            return False

    def get_info(self):
        return '{} {}: {}'.format(self.first, self.last, self.age)

    def get_email(self):
        return '{}.{}@email.com'.format(self.first, self.last)
```

Figure 10(i). Experiment-1-example

The first experiment contains five lessons, the example is from the fourth lesson. Similar to the first three lessons, which provide basic knowledge about unit testing and classic examples, as well as several coding practices, this lesson introduces the setUp() method and tearDown() method for students to learn initialization function and termination function. Finally, the ideal solution to the lesson could be like this:

```
class TestPerson(unittest.TestCase):

    def setUp(self):
        print('Test start.')
        self.p1 = Person('Bill', 'Gates', 15)
        self.p2 = Person('Steve', 'Jobs', 25)

    def tearDown(self):
        print('Test finish.')

    def test_get_full_name(self):
        self.assertEqual(self.p1.get_full_name(), 'Bill Gates')
        self.assertEqual(self.p2.get_full_name(), 'Steve Jobs')

    def test_is_adult(self):
        self.assertEqual(self.p1.is_adult(), False)
        self.assertEqual(self.p2.is_adult(), True)

    def test_get_info(self):
        self.assertEqual(self.p1.get_info(), 'Bill Gates: 15')
        self.assertEqual(self.p2.get_info(), 'Steve Jobs: 25')

    def test_get_email(self):
        self.assertEqual(self.p1.get_email(), 'Bill.Gates@email.com')
        self.assertEqual(self.p2.get_email(), 'Steve.Jobs@email.com')
```

Figure 10(ii). Experiment-1-example

When a student commits the test file, Github Classroom will automatically assess it and give the score. If the students can complete these five lessons, they basically access the ability to write tests for a given program code.

1.3 Results

All the methods could be automatically tested, but it is difficult to ensure these tests contain good assertions. For example, a simple addition test method can be written like the Figure 10(i):

```
def test_add(self):
    self.assertEqual(calculate.add(1, 1), 2)
    self.assertEqual(calculate.add(2, 2), 4)
    self.assertEqual(calculate.add(3, 3), 6)
```

Figure 11(i). Test method

This test code can successfully pass the automatic assessment, but these three lines of assertion statements are obviously flawed. The two parameters that need to be verified are duplicates, and the numbers they involve are not wide enough. The ideal test code should take into account different types of parameters, such as positive

numbers, negative numbers, decimals, etc. Meanwhile, assertions should contain a lot of combinations of various parameters; more assertion statements mean higher accuracy.

```
def test_add(self):
    self.assertEqual(calculate.add(-32, 54), 22)
    self.assertEqual(calculate.add(-535, 0), -535)
    self.assertEqual(calculate.add(-38.3845, -84.7484), -123.1329)
    self.assertEqual(calculate.add(283432, 3243443), 3526875)
    self.assertEqual(calculate.add(13.55, 24.44), 37.99)
```

Figure 11(ii). Test Method

As Figure 11(ii) shows, the assertion statements contain different types of parameters, including positive numbers, negative numbers, and decimals. The combinations of parameters are different, such as the addition of negative and positive numbers and the addition of both decimals. Compared to the previous test, the method involves a wider range of tests, increases accuracy and reliability. One of the temporary defects found in this loop is about grading granulation; now, the students have to pass all the tests to get full marks; otherwise, they get 0 points; I will describe this problem in detail in section 6.2.

1.4 Action

The problem about assertions might be challenging to solve because the auto-assessment function can only judge whether the test passed and how many tests passed. Still, it can not assess whether the tests are efficient or whether the test code has a good quality. So what we can do is to make students form the right way of thinking about unit testing through learning TDD and BDD.

2. Experiment - Can Github Classroom assess Test-Driven Development?

KEY LESSON	
Goal	To access student ability to write program code based on tests
Teacher provides	Description and requirements of the program
Students need to	1. Write the tests based on description 2. Complete program code

2.1 Plan

In this experiment, I was trying to teach TDD. Although developers have been applying TDD in various forms for several decades, this software development strategy has continued to gain increased attention as one of the core extreme programming practices (Janzen and Saiedian 2005). The course is not for students to just write tests first and write the code. It is for students to understand the features and logic of the program, write the tests point to the functions, then complete the program code according to the tests.

2.2 Process

In this experiment, the exercises are divided into two parts. The first part is given the tests, make students write the program code. The second part only gives the description and requirements of the program, make students write the tests, and make tests to drive the design, then complete the program. For example, in the second exercise of the experiment, students are required to write a program to shuffle an array. The description of the requirement is writing a function which will accept two integer arguments (min, max) and generate an array between min and max where:

i) min is an inclusive lower bound

ii) max is an exclusive upper bound

For instance, (1,7) should generate random integers as {1,2,3,4,5,6}, then the array should be shuffled. (Example URL: https://github.com/Eddie-ChenhaoLi/Exercise-02-ArrayShuffle)

If students learned the knowledge of Test-Driven Development, including the principles of TDD, the development cycle of TDD, and the design patterns of TDD in the previous lessons and completed all the exercises, they should be able to know that the program consists of two steps and can be carried out with two tests which are generating the array and shuffle the array.

```python
class TestShuffle(unittest.TestCase):

    def test_generate_array(self):
        self.assertEqual(shuffle.generate_array(2, 5), [2, 3, 4])
        self.assertEqual(shuffle.generate_array(4, 8), [4, 5, 6, 7])

    def test_shuffle_array(self):
        random.shuffle = mock.Mock(return_value=[3, 2, 4])
        self.assertEqual(shuffle.shuffle_array([2, 3, 4]), [3, 2, 4])
        random.shuffle = mock.Mock(return_value=[6, 5, 4, 7])
        self.assertEqual(shuffle.shuffle_array([4, 5, 6, 7]), [6, 5, 4, 7])
```

Figure 12. Experiment-2-Test methods

It is not difficult to come up with the first test, which receives two integer arguments and generates the array, but the second test contains a type of unique element. In fact, the Mock module is the knowledge introduced in the fifth lesson of the first teaching experiment. The author connects certain knowledge through different programs, not only to make students more proficient in the use of certain modules and functions but also to make the preceding and following courses more relevant. After completing the tests, students could clearly know the parameters and output results of the two methods, which can help students conceive the skeleton of the program code.

```python
def generate_array(a, b):
    arr = []
    for i in range(a, b):
        arr.append(i)
    return arr


def shuffle_array(arr):
    return random.shuffle(arr)
```

Figure 13. Experiment-2-Program code

2.3 Results

Students are able to write tests and program code; the idea and logic about the tests vary considerably from person to person (Antoy and Hanus 2002). Two students who have both passed the test may actually have completely different understandings of the program and may have completely different code structure and code quality. So it is worth exploring whether students can acquire an ability that can help students

shape appropriate programming logic in their minds.

2.4 Action

Each student has his own different design ideas; learning BDD may allow students to build a clearer design thinking and programming logic. BDD is an extension of TDD, but its core focus is design. In BDD, the main job is to define the behavior of the program, and the description of the behavior is the standard for testing. So we need to use a common language to write a story and the essence of the story is design thinking (Lübke and van Lessen 2016).

3. Experiment - Can Github Classroom assess Behavior-Driven Development?

KEY LESSON	
Goal	To access student ability to write Behave step files and tests based on BDD scenarios
Teacher provides	BDD scenarios
Students need to	1. Write step files based on scenarios 2. Write unit tests 3. Complete program code

3.1 Plan

In this experiment, I was trying to make students learn the design concept of BDD and learn the framework of Behave, which implements BDD. Students need to have the ability to transform natural language into a programming language. I also tried to find out how the Github Classroom supports both formative assessment and summative assessment.

3.2 Process

In the first two exercises, a scenario written in Gherkin language and the step file are given, students need to understand the program logic described by natural language in the scenario and step file, then write the tests. In the second two exercises, the only scenario is given, students need to write the step file by themselves according to scenarios and complete the tests and program code. A classic board game ‘Snakes

and Ladders' is designed in one of the exercises (Example URL: https://github.com/Eddie-ChenhaoLi/Exercise-09-SnakesAndLadders/):

```
  Scenario Outline: move
    Given I have a position <p>
    When the dice rolls out of <i>
    Then I should move to new position <np>

    Examples: move
      | p  | i | np |
      | 1  | 6 | 7  |
      | 7  | 5 | 12 |
      | 12 | 4 | 16 |

  Scenario Outline: ladders
    Given I have a position <p>
    When the position is on the bottom of a ladder <bottom>
    Then I should move to the top <top>

    Examples: ladders
      | p  | bottom | top |
      | 12 | 12     | 25  |
      | 2  | 2      | 16  |
      | 5  | 5      | 20  |
```

Figure 14. Experiment-3-Scenarios

The Behave feature file contains four scenarios which are Move, Ladders, Snakes and Win, two of them are shown in the Figure 14. The Given, When and Then statements in the four scenarios described the conditions, execution process and the results of each method. The 'Examples' table shows the values need to be tested, including inputs and outputs.

```
@Given('I have a position {p:d}')
def step_impl(context, p):
    context.p = p


@When('the position is on the bottom of a ladder {bottom:d}')
def step_impl(context, bottom):
    context.bottom = bottom


@Then('I should move to the top {top:d}')
def step_impl(context, top):
    new_position = move_ladders(context.p)
    assert new_position == top
```

Figure 15. Experiment-3-Step file

The Figure 15 shows one of the four step files, which is the Ladder step file. In essence, the step file process is similar to the unittest process. The parameters to be tested are input through the Given statement and When statement, and finally verified through the Then statement.

```
5   class TestDemo(unittest.TestCase):
6
7       def test_move(self):
8           self.assertEqual(move(1, 6), 7)
9           self.assertEqual(move(7, 5), 12)
10          self.assertEqual(move(12, 4), 16)
11
12      def test_ladders(self):
13          self.assertEqual(move_ladders(12), 25)
14          self.assertEqual(move_ladders(2), 16)
15          self.assertEqual(move_ladders(5), 20)
16
17      def test_snakes(self):
18          self.assertEqual(move_snakes(35), 23)
19          self.assertEqual(move_snakes(28), 21)
20          self.assertEqual(move_snakes(22), 15)
21
22      def test_win(self):
23          self.assertEqual(win(28), 0)
24          self.assertEqual(win(36), 1)
25          self.assertEqual(win(40), 1)
```

Figure 16. Experiment-3-Test method

Each test method uses three assertion statements to verify the parameters, and more statements mean higher accuracy. After completing the step files and test files, the skeleton of the program has become clearer, the program code could be completed based on it.

```python
def move_ladders(position):
    for bottom, top in ladders.items():
        if position == bottom:
            return top
    return position

def move_snakes(position):
    for head, tail in snakes.items():
        if position == head:
            return tail
    return position

def move(position, i):
    position += i
    return position

def win(position):
    if position >= 36:
        return 1
    else:
        return 0
```

Figure 17. Experiment-3-Program code

It should be noted that in the process of introducing the Behave framework, the automatic testing of step files needs to be set up, and the Behave framework references the Python package setuptools, so the package setuptools should be installed before executing the behave command.

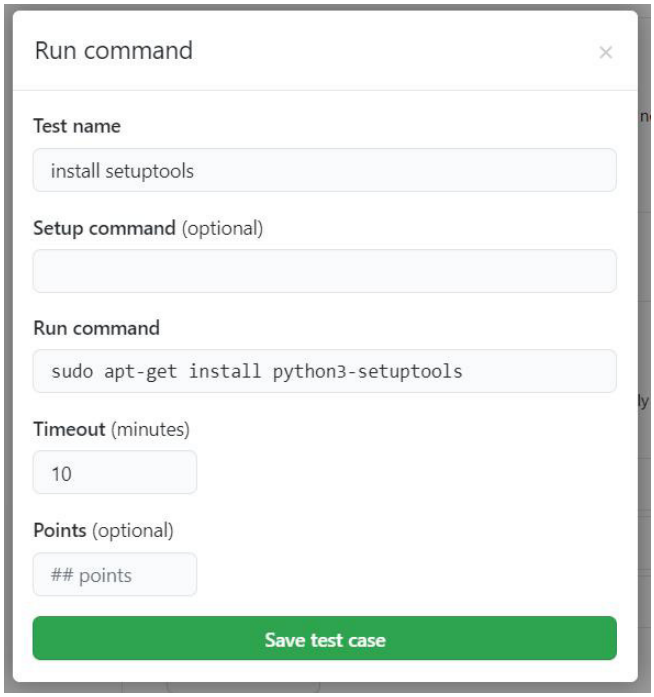

Figure 18. Test command

3.3 Results

The scenarios only described the process of primary methods; they are not related to each other and did not provide an overall workflow of the program. Therefore, it is difficult for teachers to judge whether the students have a deep understanding of the core information to be conveyed in this teaching experiment, which is using the scenarios to drive the design. Some will do it as per the lecturer's definition of correct and use it as a launchpad for learning while others will cut corners to just get it done and make the tests pass. As Biggs and Collis (1982) said, teachers could distinguish between well-learned and poor-learned by discriminates mature and immature thoughts of students. The performance varies according to its content and nature, and more subtly, in the way students perceive their performance, its importance to them, and what constitutes an acceptable level of performance to them (Biggs 1987). So teachers should pay more attention to the depth of students' learning, and also focus on the level to which students attach importance to it.

For a single assignment, Github Classroom is suitable for summative assessment. If increasing the difficulty of a single assignment, it even could be a programming quiz or exam. For a series of exercises, Github Classroom is suitable for formative assessment. Students can start learning from simple practice, then gradually be able to complete challenging exercises.

3.4 Action

Incorporating the idea of object-oriented could help the program encapsulate the objects which need to be tested. The main idea of object-oriented programming is to decompose each transaction that constitutes a problem into multiple objects. The purpose of creating objects is not to complete a step, but to describe the behavior of an object in the entire process (Lutz 2013). Object-oriented thinking is more in line with the way humans understand and think about problems, which can help students better understand and describe program logic (Castagna 2012).

4. Experiment - Can TDD and BDD drive good object-oriented design of code?

Goal	To access student ability to write object-oriented code driven by tests
Teacher provides	Code with blanks or errors
Students need to	1. Fill in the blanks in the code 2. Correct errors in the code

4.1 Plan

In this experiment, I was trying to teach writing good object-oriented code driven by tests. In the previous exercises, a scenario points to a single method, so the program written through the scenario and step file is based on the method, and multiple methods form the flow of the program. After introducing object-oriented, the scenario only describes the relationship between different objects, so the code is constructed based on the objects (Kim et al. 2001). The form of practice is also considered in this experiment; filling in the blanks and correcting errors might be two good ways for learning.

4.2 Process

The course is similar to the previous one, but the subject in the scenario is an object of the program, and all steps of the scenario are for a certain object. All the objects in the program are encapsulated and need to be imported to step files and test files. Some exercises are presented to students by filling in the blanks; they need to complete the code after the '#TODO' comment to pass the tests according to requirements and starter code. This method can improve students' ability of understanding code. Some exercises give the complete code with errors, students need to correct errors and pass the tests. This method can improve students' ability to locate and correct errors. The previous exercise 'Snakes and Ladders' is restructured in this experiment; only one scenario is given in the starter code (Example URL: https://github.com/Eddie-ChenhaoLi/Exercise-12-SnakesAndLadders):

```
Scenario Outline: move
  Given the game have a player in position <p>
  When the dice rolls out of <i>
  Then the player should move to <np>

  Examples: move
    | p | i | np |
    | 0 | 1 | 1  |
    | 0 | 2 | 16 |
    | 0 | 3 | 3  |
    | 1 | 1 | 16 |
    | 1 | 2 | 3  |
    | 1 | 3 | 4  |
    | 2 | 1 | 3  |
    | 2 | 2 | 4  |
    | 2 | 3 | 20 |
    | 8 | 2 | 10 |
    | 8 | 3 | 11 |
    | 8 | 4 | 25 |
    | 10 | 2 | 25 |
    | 10 | 3 | 13 |
    | 10 | 4 | 11 |
```

Figure 19. Experiment-4-Scenario

Compared with the previous, the four scenarios have been merged into one while the four step files were also merged, the main objects of the scenario were changed to be the game and the player, which is more realistic.

4.3 Results

To some extent, as long as the inputs and outputs of the program are correct, it can pass the test, so the automatic test cannot guarantee whether the student has written a suitable object-oriented code. The incorporating of object-oriented makes this assignment more need the intervention of the teacher to see whether the student understands and learns to use test-driven to write high-quality program code. However, reviewing all students' assignment is a heavy workload for teachers.

4.4 Action

The intervention of manual evaluation is necessary, from which the teacher can obtain more information, draw more conclusions, have a deeper understanding of the student's situation, and the students can also get more specific feedback (Saikkonen et al. 2001). In order to reduce the workload of teachers and improve the quality of teaching, a way of coordinating manual and automatic methods needs to

be designed.

5. Experiment - Can manual coordinate with automatic?

Goal	To balance manual and automatic assessment of the course
Teacher provides	A collection of 5 practice and 2 advanced exercises
Students need to	1. Complete practice collection and get full marks 2. Complete advanced exercises

5.1 Plan

In traditional courses or some online teaching systems, teachers may spend a lot of time reviewing students' work. By contrast, if the teacher only set a small number of assignments, students may not get enough practice. In the previous experiments, all the assignments could be automated assessed, including unit testing and Behave testing, which may cause cheating. At the same time, the teacher also needs to manually confirm whether there is a low-quality code that can pass the tests. So in this experiment, I tried to set up a course to balance automatic evaluation and manual evaluation, which not only allows students to fully practice programming but also reduces the burden on teachers.

5.2 Process

Chatley (2017) raised an idea in his research on improving Software Engineering Education, which is a coverage threshold. He gives each submission a score out of 5; meeting one of the requirements will get 1 point, and when students get 5 points, there will be a manual marker to confirm their assignment. The idea inspired me to design the experiment. I set a collection of 5 practice and two advanced exercises, each practice worth 100 points. Only when students complete the practice collection and get full marks, their advanced exercises will be manually reviewed by lecturers. The advanced exercises will also be automatically tested; based on it, teachers could review every submission on the Review page of the Github Classroom, and can also see all the code change on the File Change page, and even leave a comment on a single line of code. Besides, if the teacher wants to download all students' assignments, a tool named Classroom Assistant could be easily used to manage all the repositories. But in this experiment, the configuration of the automatic grading needs to be modified, which is splitting the auto-grading command to 5 separate

commands to make the unittest point to a single test file. (Example URL: https://github.com/Eddie-ChenhaoLi/Exercise-14-PracticeCollection)

5.3 Results

The necessity of adding basic tests like practice collection is beyond doubt; the first principle is 'time-on-task', the time students spend studying, which is the best indicator of student achievement (Gibbs 1999). So a series of short, templated, well-documented exercises could provide the 'time-on-task'. There is strong empirical evidence of a direct relationship between time allocation by courses, student time management, and actual student time on task, on the one hand, and student achievement on the other (Gibbs and Simpson 2004). If the students have to pass, then they have to spend some time doing the work, and the hope is that in so doing, they will be interested enough to get some learning and will then want to spend more time developing their understanding.

Although automatic evaluation could improve efficiency in various aspects, it is unlikely that manual evaluation will be replaced entirely. The manual assessment could find some logic vulnerabilities, which are hard to discover by automatic. Besides, the tools or system of automatic might need to be updated frequently. Still, humans can immediately update their knowledge and techniques; human judgment and intuition always benefit the manual elements. In fact, automation and manual are not opposites; they have their own advantages, they can complement each other and will exist at the same time for a long time (Higgins et al. 2005). Besides, as mentioned in 3.3, it is easy to assess students' code, but the level of students' learning is difficult to be evaluated. Evaluation should not just produce a result but should perceive the gap between where they currently are and where they should be (Biggs 1998). The combination of automated tests and manual review would be an excellent way for both teachers and students to use the Github Classroom teaching system. Teachers do not need to spend a lot of time to review the basic practice while students could actually improve their programming skills.

6. Still Unknown

In the series of teaching experiments, most of the needs are satisfied by many advanced functions provided by Github Classroom, but there are still some problems that cannot be solved temporarily. Github Classroom is still under development, and these problems may be updated in the future or there will be compensatory functions.

6.1 Student operation permissions

In the current version of Github Classroom, students have operation permissions on all files in their repositories so that students may tamper with test files and template codes. If Github Classroom updates the function of modifying the permissions of repository files in a future version, then the teacher can better control the file permissions to implement some functions. For example, the teacher can give some template codes so that the students can only fill in the blank parts and cannot modify the existing codes. At the same time, the function can prevent students from tampering with the test code to pass the tests.

6.2 Automatic grading granulation

One of the features being developed by Github Classroom is automatic grading granulation. According to the Github Classroom Program Manager Woodthorpe (2020), the current defect is that when students pass all test methods in a test file, they can get full marks, and when a single test fails, they will get 0 points. So the managers need to break their testing suite into different GitHub Classroom tests and assign each one a point value.

7. Future work

After conducting the teaching experiments of unit testing, TDD and BDD, the learning course has become almost complete. There are still a lot of possible directions worth exploring. Specific experiments or lessons without Github Classroom may provide opportunities for students to improve their technical capabilities in different dimensions. In this chapter, I am going to introduce five interesting future works, including creating BDD scenarios, learning testing tools, learning integration testing,

etc.

7.1 Creating BDD scenarios

In the previous experiments, the BDD scenarios were given for students to read and understand the logic of the program, so they do not have the opportunity to learn and write the scenarios. If students learn and master the ability to use Gherkin language and write scenarios, they can smoothly describe the workflow of a program or the execution of a method in natural language. The significance is that in future group development or enterprise-level development, the scenarios and descriptions could even be understood by non-programmers such as test engineers, product engineers, etc.

There might be two or more forms of the teaching experiment:

i. Filling the blanks

Set a problem according to actual product development. Among the three basic element words, the teacher may give the GIVEN and WHEN statements and make students complete the THEN statement. With difficulty increasing, the GIVEN statement is given, and students need to write the WHEN and THEN statements. Another possible way is giving one of the statements with blanks, make students complement missing conditions or elements. Finally, after finishing all the statements, they can get a complete scenario and compare the problem with the scenario. For example, given a simple scenario with a blank:

Scenario: using a squeezer

GIVEN I have an apple and a squeezer

WHEN _____

THEN I get apple juice

Students are required to fill in the blank of the WHEN statement. According to the description of the scenario, which is ‘using a squeezer’ and the GIVEN and the THEN statement, the WHEN statement should be ‘WHEN the apple is put into the squeezer’. This kind of classic practice allows students to learn to use natural language to describe a problem or an event.

ii. Writing analysis report

The experiment is suitable for open teaching with students writing how do they understand and analyze the problem, organize the process of the program, and finally complete the BDD scenario in a report. So Github Classroom may not be able to provide much help for it, nor can it support automatic assessment. This kind of report should be manually reviewed by the teachers. For example, when the teacher gives an event about registering a Github account, students should analyze how many elements are involved in this event, what conditions were triggered, and what results were obtained in the end. We could know the basic process is that we need to open the registration page, enter the information required, and click the registration button (Holmes 2020). So the scenario could be:

Scenario: register account

GIVEN I am on the registration page

WHEN I enter the information required

AND I click the registration button

THEN the account is created

The scenario briefly covers the principal elements and the process. Students need to understand the problem and write down the process of analyzing the problem, organize the elements and processes of the event, and finally write the scenario. The report and the scenarios could be written in readable files such as pdf, md files, etc. And what the Github Classroom could help is that all kinds of files could be uploaded directly to the repository, so the lecturers can have only one source of submission, which helps them better manage resources.

7.2 Improving code quality

SonarQube is an open platform for managing code quality, which can quickly locate potential or apparent errors in the code. The advantage is that you can see the situation of project code through an excellent graphical interface, and you can also find deep design flaws such as memory leaks, repetition rates, or code complexity (Campbell and Papapetrou 2013). The fundamental reason for program errors is that there are problems with code quality, and it is necessary for students to use tools to improve code quality. The testing tool can be a self-study course for students, and

there are plenty of materials and tutorials on the Internet.

In addition to using tools, some forms of teaching can also make students pay more attention to code quality, such as a code coverage test. Developers and testers use code coverage to ensure that all or substantially all statements in a program have been executed at least once during the testing process (Tikir and Hollingsworth 2002). A basic way is to have students try to write a test with 100% coverage; for a simple example, this could be based on code that implements a game like Rock Paper Scissor, and students need to test all the possibilities. For another example, the students would check that 7,7,7 beats 10,10, but that 10, Ace beats 7,7,7, etc. in a game of Black Jack. But in fact, code coverage does not completely represent the quality of the tests; the students need to consider the pros and cons of test design and its relationship with code quality in combination with code coverage (Wong et al. 2007). The lecturer could provide a good, readable implementation and a poor, unstructured, hard to read version and split the class, which would then lead to a discussion on code quality.

7.3 Learning integration testing

The integration testing refers to that on the basis of completing unit testing, assemble all the modules into a system according to design requirements and design structure to conduct a combined test (Sinyagin et al. 2016). In fact, although some modules can work individually, there is no guarantee that they can normally work when combined, so the quality of a single module is not enough to reflect the quality of the overall operation.

7.4 Learning use case

In software and systems engineering, a use case is a list of actions or event steps typically defining the interactions between a role and a system to achieve a goal. In order for students to have the ability to make a scientific organization of the behavior of software testing, students need to learn to write test cases, including the input of tests, test execution conditions and expected results (Swain et al. 2010).

7.5 Advanced features of Behave

In the previous experiments, only the basic functions of Behave were involved. If students are required to be proficient in using the Behave framework, some experiments and exercises could be set up to make students learn advanced functions. For example, it will be quite helpful for students to learn Environmental Control of the Behave. The environment.py module may define code to run before and after certain events during the testing. A common use-case for environmental controls might be to set up a web server and browser to run all tests (Engel et al. 2019).

Advantages

In this chapter, I will introduce the advantages of Github Classroom compared to other teaching systems and traditional education. I think these advantages of Github Classroom can help teachers and students improve efficiency and learning quality. Github Classroom may become a more mainstream teaching system that is widely used by high schools and universities.

1. Easy to set up

1.1 Classroom

Classrooms are easy to set up, lecturers could set different courses or different stages of the same course as different classrooms, making it easy to manage courses and exercises.

In a traditional teaching process, lecturers might give a rule of naming files to manage students, which is usually a combination of name, ID, and marks. But wrong naming still often appears. A roster, which is a CSV or text file, could be used to list all the students, which helps count the situation of students' assignments. The only job is that students need to connect their Github accounts to the roster.

1.2 Assignment

Assignments are easy to set up; lecturers only need to set the title, deadline,

template code, and tests, then send a link to invite students to accept.

In a traditional teaching process, lecturers have to send the starter code to students through office software or email, and students have to download it.

In comparison, creating assignments and sending the invitation, instead of sending the whole project, the former would be more rational and faster.

1.3 Group Assignment

Github Classroom can easily implement group cooperation tasks by using group assignments. With the system, all groups of students could be easily assigned to teams by lecturers and receive their own Github repositories. Then the students could collaborate on a project using Git and Github, which is what they would learn and do in an academic or industry research project. The contribution of each student to the project could be easily seen by lecturers since all the commit histories of students are accessible to the lecturers (Fiksel et al. 2019). Furthermore, it is unnecessary for teachers to worry about the budget or resource limitations because all kinds of assignments could be created freely and infinitely, and the quantity the size of repositories is unlimited.

2. Code management

2.1 Version control

Within the deadline, students can update their version at any time to modify the code or correct errors. Every single version of code updated by students is saved and could be seen by lecturers. Through the process of managing code, students are able to learn and even could be proficient in using Git and Github, which not only improves their own ability of version control but also enhances the capability of code management when working in a team. However, from my own experience, version control is not included as an independent course by the university; it is regarded more as a common tool so that students do not pay much attention to it and do not have opportunities to practice. As a matter of fact, there are implicit benefits to the use of Git and Github in actual corporate work, including enhancing employability, supporting remote development or offline development, and even supporting parallel development in large scale software.

2.2 Repository management

Each assignment is an independent project, this structure is convenient for management. Lecturers can directly download all students' repositories, a structured catalog based on student information and assignment information could be automatically generated by using the tool 'Classroom Assistant'.

3. Auto assessment

3.1 Automated test

The automated tests are instantaneous and completed within 10 seconds. Every time a student pushes the code, it will be automatically tested. Students could know what went wrong and make corrections. All tests are executed in a virtual environment, do not occupy any resources, and do not need to install anything.

3.2 Auto-grading

Assignments are auto-graded by tests, which directly reflect the accomplishment of students. The temporary defect is that the points are based on grading commands instead of testing methods, so the points per test are not very granular. It is still under development, according to a Github Classroom staff.

4. Feedback loop

4.1 Feedback page

In learning, it is essential that students be given the opportunity to practice their new programming skills in an environment where they can receive constructive and corrective feedback (Venables and Haywood 2003). The feedback page is a perfect platform for lecturers and students to communicate. Students could see teachers' evaluations, while the code is next to it for comparison. Teachers and students could also have an instant messaging chat on the page, which is more convenient and effective than using email.

4.2 Code review

Most pedagogical exercises may consist of hundreds of lines of code, and some

actual programs exceed thousands of lines; any line of code error may cause problems in the program. Whether it is face-to-face teaching or online teaching, the code guidance is complicated and difficult both for teachers and students. The Github Classroom provided possibilities to make code guidance easier; teachers can even evaluate single lines of code by reviewing files changes and leave a detailed comment on the particular line of code, which is a very targeted guidance for students. This kind of function of reviewing code and giving advice is difficult to achieve in traditional teaching.

4.3 Commit report

Teachers could also directly submit the documents such as a report, test file, or tutorial documentation to students' repositories, which is almost the fastest way a teaching system can do; there is only one step needed to achieve the file transfer. The traditional way is that lecturers send each file to students as an attachment through an office software or email, which is inefficient.

Limitations

Although Github Classroom has many outstanding features, it is still not a perfect platform and has some limitations. Being able to find and solve these defects will make the teaching system more helpful. I will introduce some problems I found, and certain possible solutions are given.

1. Lack of interaction

Students' learning outcomes depend largely on the teacher's interaction and efficiency in the classroom (Hamre et al. 2013). In the process of completing online assignments, there is a lack of face-to-face communication between students and lecturers. To some extent, enriching the forms of assignments and formats of materials could possibly compensate for the problem. For students, a video recording about program design or problems encountered could definitely become supplement to the code and tests. In addition, teachers could include some sort of slide materials

using reveal.js or a presentation using framework impress.js to change the dynamic on the feedback. Github allows the transfer of any file, so any format such as text, sound, animation, etc. may change the way of communication between students and teachers. The author encourages any form of innovation to improve interaction.

If Github Classroom is considered to be a complete course, lecturers can record a tutorial video or write tutorial documentation as a guide and communicate with students frequently during the process of feedback. If Github Classroom is considered to be a supplement, it could be used to do programming exercises after the traditional lecture. In-class education can allow teachers to notice the students' potential struggle, assess the current situation, and provide direct feedback; online courses also need a scaling solution to detect students in trouble (Teusner et al. 2018). Harmonizing the two methods may become one of the best ways of modern education.

2. Cheating

2.1 Plagiarism

Since Github Classroom relies on repository management, some students might copy the code from others. If solutions are given in repository as reference, some students might copy the solutions to pass the tests. If the Github Classroom is used mainly for formative, then this is not a serious problem, but with summative assessment, the issue still needs more exploration to be perfectly solved. Plagiarism may not be accurately detected, but according to a contributor of Github Education Community Wallach (2020), it appears that the Github 'events API' provides a series of time-stamps along with the Git commits ids, so the time-stamps could be received when the commits actually hit the Github server, which could be used to detect when each student pushes their work to the server. Matthias (2018) said, MOSS and script to clone repositories is worth investigating which helps analyzing students' work. Another way is to have students provide an explanation for the code. Like Topher (2019) said, assessments could emphasize the ability to clearly express algorithms, solve problems conceptually, design the architecture of a solution. Students always realize that they have to be able to explain their design decisions

and algorithmic mechanics, which reduce sense of ‘police-vs-code-copiers’.

2.2 Tampering code

Since automated tests are carried out through test files in repository, so there is a risk that some students might delete the test code or modify it to a simple test to make tests pass. The problem might be temporary, according to a Github Staff Woodthorpe (2020), GitHub Classroom is planning on solving this by saving the tree sha of the ‘.github/’ directory when we setup autograding. Because of how the Git FS works like a Merkle tree, any change to the autograding files will change the tree sha.

3. Rationality of exercises

Students’ active participation of curriculum design is considered by more and more universities, most of the feedback are seen helpful (Brooman et al. 2015). Some learners obviously need more guidance and practice than others; instead, challenging tasks should be set up to attract motivated and capable students (Warren 2002). If a Github Classroom course is set up by the lecturer from the beginning, it might lack feedback from students on the difficulty of assignments and rationality of the lesson.

Conclusion

The research has shown almost all the available functions and the entire workflow both from teacher and student’s view, which demonstrates that the Github Classroom simplified most of the operation steps compared to other teaching systems, and much more convenient than the traditional teaching. The process of using Github Classroom to teach Python unit testing, TDD and BDD is provided, which proves that the system could be used to teach other courses. All the descriptions and diagrams are good guidance for teachers or students who want to set up an online learning course or who want to learn and use Github Classroom.

I strongly recommend teachers who want to reduce their workload of assessing

students' learning outcomes to use Github Classroom. It is also one of the best options for teachers who want increase the amount of students' practice and improve the programming skills of students. It is convenient to establish a programming language course such as Java, Python, Node, and C because the testing of these languages is nicely supported by the auto-grading function of Github Classroom. Automatic assessment, quick feedback, and other functions I mentioned in the previous sections are the reasons why Github Classroom is competitive. Various computing courses could be smoothly constructed by making full use of these advantages.

During the research, I learned about various popular learning system which has its own advantages and limitations. I knew most features and functions of Github Classroom and learned the complete workflow of it. Before setting up the course, I have to read and understand plenty of materials about unit testing, TDD, and BDD, which allows me to learn a lot about testing in depth. The most important is, as the first time to solve a problem from a teacher's perspective, I experienced a completely different way of thinking. I tried to design the lessons and exercises to be interesting and easy to understand while avoid writing the tutorial documentations to be tedious. What I always think about is how to make it easier for teachers and students to communicate. All these different perspectives of experience and perception have had an impact on me, which improves my ability to teaching and makes me always think from different positions in my future learning and work.

At the end of the article, when I put my perspective on the relationship between higher education and educational tool or suppliers, I would like to put forward some thoughts about the future. On the one hand, now the Github has become the largest code hosting platform, and Github Classroom is becoming more popular based on advanced features; meanwhile, Visual Studio Codespaces, a cloud IDE, is integrated into Github. The situation means that using Github Classroom is might just the start of a more significant relationship with Microsoft. Ideally, the supplier can provide a complete and feature-rich system from various positions, including education, source control, and online development. But on the flip-side, it might be a risk to be locked

into one big vendor. When the supplier has a large impact on the educational system, it might cause damage to the innovation and independence of higher education. So there are a lot of questions worth thinking about, are there good alternatives such as Gitlab? Can universities develop their own teaching system based on Git alone? Would the university gain most of the benefits of Github Classroom using its own Gitlab server? I think that universities might need to find a way to make good use of tools and resources while maintaining the essence of education itself.

Reference

Ala-Mutka, K.M., 2005. A survey of automated assessment approaches for programming assignments. Computer science education, 15(2), pp.83-102.

Antoy, S. and Hanus, M., 2002, September. Functional logic design patterns. In International Symposium on Functional and Logic Programming (pp. 67-87). Springer, Berlin, Heidelberg.

Beck, K., 2003. Test-driven development: by example. Addison-Wesley Professional.

Biggs, J., 1998. Assessment and classroom learning: a role for summative assessment?. Assessment in Education: Principles, Policy & Practice, 5(1), pp.103-110.

Biggs, J.B. and Collis, K.F., 1982. Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome). Academic Press.

Biggs, J.B., 1987. Student Approaches to Learning and Studying. Research Monograph. Australian Council for Educational Research Ltd., Radford House, Frederick St., Hawthorn 3122, Australia..

Black, P. and Wiliam, D., 2009. Developing the theory of formative assessment. Educational Assessment, Evaluation and Accountability (formerly: Journal of Personnel Evaluation in Education), 21(1), p.5.

Brooman, S., Darwent, S. and Pimor, A., 2015. The student voice in higher education curriculum design: is there value in listening?. Innovations in education and teaching international, 52(6), pp.663-674.

Campbell, G. and Papapetrou, P.P., 2013. SonarQube in action. Manning Publications Co..

Castagna, G., 2012. Object-Oriented Programming A Unified Foundation. Springer Science & Business Media.

Chatley, R. and Field, T., 2017, May. Lean learning-applying lean techniques to improve software engineering education. In 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET) (pp. 117-126). IEEE.

Engel, J., Rice, B. and Jones, R., 2019. Behave Tutorial. [online] Behave Documentation. Available at: <https://behave.readthedocs.io/en/latest/tutorial.html> [Accessed 2 September 2020].

Fiksel, J., Jager, L.R., Hardin, J.S. and Taub, M.A., 2019. Using GitHub Classroom To Teach Statistics. Journal of Statistics Education, 27(2), pp.110-119.

Gibbs, G. and Simpson, C., 2004. Does your assessment support your students' learning. Journal of Teaching and learning in Higher Education, 1(1), pp.1-30.

Gibbs, G., 1999. Using assessment strategically to change the way students. Assessment matters in higher education, 41.

Hamre, B.K., Pianta, R.C., Downer, J.T., DeCoster, J., Mashburn, A.J., Jones, S.M., Brown, J.L., Cappella, E., Atkins, M., Rivers, S.E. and Brackett, M.A., 2013. Teaching through interactions: Testing a developmental framework of teacher effectiveness in over 4,000 classrooms. The Elementary School Journal, 113(4), pp.461-487.

Harlen, W. and James, M., 1997. Assessment and learning: differences and relationships between formative and summative assessment. Assessment in Education: Principles, Policy & Practice, 4(3), pp.365-379.

Higgins, C.A., Gray, G., Symeonidis, P. and Tsintsifas, A., 2005. Automated assessment and experiences of teaching programming. Journal on Educational Resources in Computing (JERIC), 5(3), pp.5-es.

Holmes, R., 2020. Writing BDD Test Scenarios - Department Of Product. [online] Department of Product. Available at: <https://www.departmentofproduct.com/blog/writing-bdd-test-scenarios/> [Accessed 10 September 2020].

Ihantola, P., Ahoniemi, T., Karavirta, V. and Seppälä, O., 2010, October. Review of recent systems for automatic assessment of programming assignments. In Proceedings of the 10th Koli calling international conference on computing education research (pp. 86-93).

Janzen, D. and Saiedian, H., 2005. Test-driven development concepts, taxonomy, and future direction. Computer, 38(9), pp.43-50.

Kim, S.W., Clark, J.A. and McDermid, J.A., 2001. Investigating the effectiveness of object‐oriented testing strategies using the mutation method. Software Testing, Verification and Reliability, 11(4), pp.207-225.

Loeliger, J. and McCullough, M., 2012. Version Control with Git: Powerful tools and techniques for collaborative software development. " O'Reilly Media, Inc.".

Lutz, M., 2013. Learning python: Powerful object-oriented programming. " O'Reilly Media, Inc.".

Lübke, D. and van Lessen, T., 2016. Modeling test cases in BPMN for behavior-driven development. IEEE software, 33(5), pp.15-21.

Matthias, M., 2018. Github Classroom: Plagiarism & Cheating Detection. [online] Github Education Community. Available at: <https://education.github.commu

nity/t/github-classroom-plagiarism-cheating-detection/11859> [Accessed 18 September 2020].

Moen, R. and Norman, C., 2006. Evolution of the PDCA cycle.

Runeson, P., 2006. A survey of unit testing practices. IEEE software, 23(4), pp.22-29.

Saikkonen, R., Malmi, L. and Korhonen, A., 2001, June. Fully automatic assessment of programming exercises. In Proceedings of the 6th annual conference on Innovation and technology in computer science education (pp. 133-136).

Sinyagin, A., Vinall, J.P. and Wang, F., Vertafore Inc, 2016. Integration testing method and system for web services. U.S. Patent 9,367,435.

Soeken, M., Wille, R. and Drechsler, R., 2012, May. Assisted behavior driven development using natural language processing. In International Conference on Modelling Techniques and Tools for Computer Performance Evaluation (pp. 269-287). Springer, Berlin, Heidelberg.

Swain, S.K., Mohapatra, D.P. and Mall, R., 2010. Test case generation based on use case and sequence diagram. International Journal of Software Engineering, 3(2), pp.21-52.

Teusner, R., Hille, T. and Staubitz, T., 2018, June. Effects of automated interventions in programming assignments: evidence from a field experiment. In Proceedings of the Fifth Annual ACM Conference on Learning at Scale (pp. 1-10).

Tikir, M.M. and Hollingsworth, J.K., 2002. Efficient instrumentation for code coverage testing. ACM SIGSOFT Software Engineering Notes, 27(4), pp.86-96.

Topher, M., 2019. Github Classroom: Plagiarism & Cheating Detection. [online] Github Education Community. Available at: <https://education.github.communit

y/t/github-classroom-plagiarism-cheating-detection/11859> [Accessed 19 Septemb er 2020].

Venables, A. and Haywood, L., 2003, January. Programming students NEED instant feedback!. In Proceedings of the fifth Australasian conference on Computing education-Volume 20 (pp. 267-272).

Wallach, D., 2020. Github Classroom: Plagiarism & Cheating Detection. [online] Github Education Community. Available at: <https://education.github.communit y/t/github-classroom-plagiarism-cheating-detection/11859> [Accessed 5 Septembe r 2020].

Warren, D., 2002. Curriculum design in a context of widening participation in higher education. Arts and Humanities in Higher Education, 1(1), pp.85-99.

Wong, W.E., Qi, Y., Zhao, L. and Cai, K.Y., 2007, July. Effective fault localization using code coverage. In 31st Annual International Computer Software and Applications Conference (COMPSAC 2007) (Vol. 1, pp. 449-456). IEEE.

Woodthorpe, N., 2020. Github Classroom Autograding / Testing. [online] Githu b ClassroomCommunity. Available at: <https://education.github.community/t/gith ub-classroom-autograding-testing/52304/11> [Accessed 8 August 2020].

Woodthorpe, N., 2020. Prevent Students From Changing Autograding Tests. [on line] Github Education Community. Available at: <https://education.github.comm unity/t/prevent-students-from-changing-autograding-tests/53718> [Accessed 12 S eptember 2020].