

# report

*by* Zhe Wang

---

|                |                                                      |                 |       |
|----------------|------------------------------------------------------|-----------------|-------|
| FILE           | 17853_ZHE_WANG_REPORT_227501_161477270.DOCX (25.39K) |                 |       |
| TIME SUBMITTED | 17-JAN-2020 12:35PM (UTC+0000)                       | WORD COUNT      | 3551  |
| SUBMISSION ID  | 118583638                                            | CHARACTER COUNT | 18983 |

# Report

**Name:**ZHE WANG

**Student Number:1904779**

## **TASK1**

### **1 Test-Driven Development**

#### **1.1 Definition of Test-Driven Development**

Test-Driven Development is an thought of Agile Development[1]. It is different from the traditional development mode, which is to test the code after the completion of software development. The basic idea of Test-Driven Development is that developers need to write test code before developing function code[1]. That is to say, after it is clear to develop a function, the first thing that developers should think about is how to test this function that they want to achieve, and complete the writing of test code. Then write the functional code to implement the required functionality to make it pass the test. Then loop to add other functions through adding test code and write functional code until all functions are developed. The process of test driven development is as follows.

1. The development team should know what they need to do and define functions to be implemented, which can be recorded as a TODO table.
2. Developers need to write test cases for this function quickly. At this time, developers focus on requirements, program input and output, without worrying about how to realize internal functions.
3. The test code must fail to compile. Because the functions in the test code are not written. If the test code is successfully compiled and executed, the developer needs to solve the problem.
4. Write the function code corresponding to the test code. When writing function code, developers only need to consider the implementation of this function, and do not need to consider other requirements and functions. They can achieve the current function in the simplest way, and do not need to pay attention to whether the code is efficient or not.
5. Test whether the function code can pass the test. If it fails, developers should modify it to make it success.
6. Refactoring and simplifying the code to make it cleaner and ensure that the modified code passes the tests.
7. Repeating steps 1 through 6 until all features are developed.

## **1.2 Benefits of TDD**

(1) TDD integrates testing into development, which also considers the requirement of testing. Development and testing are carried out at the same time. Developers do not need to spend a lot of time testing the feasibility of the code after finished all the functions of the software.

(2) It reduces the burden of developers. TDD provides a clear development process, which can make developers to focus on one function at a time in software development, reducing the difficulty of development. At the same time, it also increases the enthusiasm of developers with clear goals.

(3) TDD has full coverage of unit tests[2], which provide a protection network for product code. So that developers can easily meet changes in demand or improve code design. When customer requirements change, developers can quickly modify the functions needed through unit test to complete the function modification. Writing tests first helps developers think about the requirements and clarify the details of the requirements in advance, rather than half the code to find the requirements that are not clearly defined.

(4) TDD can help people to design simple, clear and easy-to-use interfaces. Because developers always need to write test code first. In order to facilitate testing, developers need to take the initiative to

design simple and easy-to-use interfaces to make testing easier.

(5) TDD can reduce the coupling degree between different functions.

Because each function is implemented in a unit test.

(6) The requirements of the software are described clearly by the test code, which can reduce a lot of unnecessary documents. TDD saves a lot of time which improves the efficiency of writing documents in development.

## **2 Behaviour-Driven Development**

### **2.1 Introduction of Behaviour-Driven Development**

BDD is a design method of agile software development based on TDD, which can make the development team develop the right product more efficiently. BDD encourages collaboration among developers, QA and non-technical personnel or business participants in software projects[3]. The common language of BDD is a kind of near natural language to describe software. Customers and developers can understand and use this language. Based on it, it can greatly simplify the communication between the customer and the development team, making it easier for the development team to better understand the customer's needs and realize the functions required by the customer, which avoid unnecessary workload due to inadequate understanding of

needs of customers. BDD uses many stories in development[3]. System business experts, developers and testers work together to analyze software requirements, and then write these requirements into stories. Developers fill in the content of these stories, and testers test the results of these stories. The practical process of BDD is as follows.

- (1) Customers put forward the demand for products.
- (2) Business departments put forward specific business target and software requirements according to customer requirements .
- (3) Product Manager (Requirement Analyst) and development team will sort out user stories together and instantiate user story acceptance conditions.
- (4) The developer (testers) use BDD tools, such as cucumber, to convert them into operational acceptance test instruction documents, which are also automatic acceptance test cases.
- (5) the developer design software functions according to user stories, write unit tests and implementation codes. When the software functions pass unit tests, it means that the software functions meet the design requirements.
- (6) The developer run the acceptance test code generated in step 4. The acceptance test can be the UI layer end-to-end test or the interface test.

(7) The product manager can see the acceptance test report in real time, which is also the product description document.

## **2.2 Advantages of Behaviour-Driven Development**

(1) BDD provides a general, simple and structured description language, which can be understood even by customers who do not understand programming. Through BDD, the development team, business personnel and customers can communicate smoothly. This way can greatly ensure that the software developed by the development team meets the requirements of customers, which can improve development efficiency.

(2) Development teams are able to deliver products faster because there is less ineffective communication between customers and development teams.

(3) BDD can save development cost. Through BDD, the development team can clearly know what customers need, so that the development team can largely develop the right functions. This reduces the likelihood that the development team will create invalid and incorrect functionality.

## **3 Feature-Driven Development**

### **3.1 Definition of Feature-Driven Development**

FDD (Feature Driven Development) is an agile software

development process. It emphasizes feature driven and fast iteration, which can ensure rapid development and software quality. Feature is a core concept of FDD[4]. Features refer to small functions that users need. Each feature must be small enough to be completed in two weeks. In addition, every feature must be the function that users need, focusing on customer value. A feature can be mapped to a step of a business activity. FDD has good enforceability and can monitor the development process of the project accurately and timely[4]. It can construct software correctly and timely. FDD also breaks the traditional barriers that separate domain and business experts / analysts from designers and implementer. Analysts are freed from the abstract work and directly participate in the system construction work of developers and users. The FDD process is as follows:

(1)Develop an overall model: This is the beginning of FDD. Generally, an object modeler will lead his experts to work together. They will build a model throughout the whole system and its internal and external relations.

(2)Build feature list: generally, the development team decomposes the functions of a software, and then divides each domain into major feature sets, and each major feature set refines the feature set. Each step in the activity can be divided into a feature, thus forming a feature table. The following formats are generally used to describe features:

For function: < action > the < result > < by | for | of | to > A (n) < Object >

For function set: < action > < ing > A (n) < Object >

For main function set: < Object > Management

(3)Plan by feature: according to the importance, difficulty and other factors of features, the development team needs to plan the order of features. Therefore, the development team should list a feature development plan to arrange the implementation sequence of features. A date by which business representatives and deployment teams can plan which characteristics to deliver. It also allows business people to understand some information about each feature, such as development and test dates.

(4)Design by feature: these features will be assigned to each programmer for implementation. They will select a set of features from the tasks assigned to them, specify the development sequence diagram for the selected features, and then based on the content of the sequence diagram, the programmer will further refine the object model and determine the class and object.

(5)Build by feature: programmers implement classes through design packages generated in (4), and then conduct unit tests and code reviews. Among them (4) and (5) are iterative processes.

### **3.2 Advantages of Feature-Driven Development**

(1) Compared with the existing methods, Feature-Driven Development makes developers pay more attention to the modules that can reflect customer value. Because Feature-Driven Development is based on features to develop function, and every feature pays great attention to customer value.

(2) FDD can easily monitor the progress of the project. FDD process is based on the feature and develops around feature. Therefore, it can provide accurate progress reports to customers, project team members and business personnel through the completion records of each feature, so as to facilitate the communication of project related personnel.

(3) Customers also can participate in the development process. They can see the development results of all previous stages of software developers, so as to determine whether they meet their own needs. If not, the development team only needs to modify the functions of the previous stage. In addition, by participating in software development, customers can also propose what functions should be completed next.

### **4. Application scenarios of TDD, BDD and FDD**

TDD is suitable for development with sufficient time. Compared with traditional development, TDD needs to write test code before function

design and implementation. It takes a lot of time. In addition, TDD is also suitable for product and platform development. On the other hand, TDD is not suitable for development in several cases. For example, if developer need to develop a product (or software module) with high customization for users, or design a product full of trial, it needs constant trial and improvement, and find out problems and solve them in the trial. In this case, developers are often not sure what specific functions the requirements and products need, so it is difficult to design the correct test code. These situations are not suitable for test driven development. BDD is suitable for some software development focusing on user experience, such as some interactive development (Facebook, National Express). BDD focuses on the effective communication between customers and development teams. Through BDD, developers will consider and design the functions required by the software from the perspective of customers, so that developers can know what customers really need, thus reducing the development of useless functions. In addition, BDD is also suitable for large project development. The development team divides a project into functions through user stories, and then completes them, which also improves the work efficiency. FDD is applicable to those systems or software with uncertain or frequently changed requirements, because in the process of FDD, the customer and the development team have intensive and effective

communication. When the customer needs to change properly, FDD ensures that the development team can understand and modify in time, and also ensures that the project team can submit valuable results faster without reducing its quality. In addition, FDD is also suitable for complex software development projects.

## **TASK2**

### **1 High Cohesion And Loose Coupling**

#### **1.1 High Cohesion**

High cohesion is a standard to evaluate the quality of software design in software engineering. Cohesion is a measurement standard of the relevance and concentration of elements in a system that need to accomplish the same task. The closer the elements are combined, the higher the cohesion. In other words, high cohesion means that the elements inside a software module are related to each other, and the elements that are not related will not exist in the module. For example, when God built a dog (when defining a class), he only gathered the characteristics of the dog together, and removed any characteristics that are not related to the dog, such as the horn of a cow, the mane of a horse.

This makes a class more pure. In addition, each method in the highly cohesive module will try to achieve a small function without calling the

methods and properties of other modules. That is to say, the principle of single responsibility.

## **1.2 Loose Coupling**

Loose Coupling in software engineering refers to the poor correlation between modules in software system. One module is less likely to call attributes or methods in other modules. There are High independence between modules. In other words, loose coupling is to let each module, as far as possible, complete a specific sub-function independently. The interface between modules is as few and simple as possible.

## **1.3 Advantages of High Cohesion and Loose Coupling**

There are three advantages of high cohesion and low coupling in software development. They are good readability, excellent reusability and program is easy to maintain and easy to rewrite.

(1) Good readability: if the principle of high cohesion and loose coupling is followed in software development, one class in the code written by developers will reduce the use of other classes. Each class has its own specific functions. There is no duplication of functions between classes. Each class can clearly understand its meaning through its name or

description, and each function of the class has only one small function. In these cases, developers or other programmers can easily understand the logic of this program and the function and implementation code of each class, which greatly improves the readability.

(2) Good reusability: in software development, developers adhere to the principle of high cohesion and loose coupling to encapsulate different functions into different classes, and each class can ensure relatively independent completion of tasks within its own scope of responsibility, while ignoring other tasks unrelated to its own functions. When a function needs to be used, developers do not need to write the same code again, but can directly use the previously written function, which is very convenient. For example, if the code to be used is in the same object, developers only need to write the code into a function and call the function when necessary. If the code to be used is in another class, it is very effective to abstract this class into an object and call the required code functions. If the code to be used is in another project, developers can make this project into a component and reference it to each project. Function reuse, object reuse and component reuse improve the readability of the code, simplify the code, and make other programmers understand the code more easily.

(3) Good maintainability and changeability: code implemented according to the principle of high cohesion and loose coupling has low correlation

between classes, and one class calls less attributes or methods of other classes. Each class only implements functions within its own scope of responsibility. When there are errors or bugs in the software, because each function is encapsulated in the corresponding class, the maintainers and developers can quickly locate the function or class that may be in error according to the type of errors. It reduces the number of codes that maintenance personnel need to check. The maintainability of the program is improved. In addition, if the software needs to be changed frequently, it can adapt to the market demand in time. High cohesion and loose coupling software minimize the scope of each change. When a function needs to be changed, developers only need to modify some functions in the corresponding class. For example, if the review form is changed, it will only be related to the object of the review form, and we will not change other objects.

## **2. Data Hiding and Information Hiding**

### **2.1 Definition of Data Hiding**

Data hiding is a method used to hide information in computer code in object-oriented programming. When defining the properties of an object, it is necessary for developers to determine the visibility of these properties. That is, which properties are visible externally and which

properties are invisible externally. Only internal functions can call these properties. Generally, other objects should be forbidden to directly access the data in an object, but these data can be accessed by other objects through the operation interface, which is called information hiding. For example, in a class, the data will not be directly exposed to other classes for access, but the private keyword is used to hide the data. If the private properties need to be accessed outside the class, the corresponding get and set methods can be provided in the class, so that users can indirectly access the private properties outside the class.

## **2.2 Advantages of Data Hiding**

(1) Data hiding improves the security of software. Through data hiding, important data can be hidden without being modified by other people, which improves the security of software, because if the data is not hidden, anyone can call any data in an object and modify it. For example, if anyone can access and change the password of a bank account. Then the bank will make a big loss.

(2) It is impossible for a programmer to accidentally link to a wrong data. Data hiding can ensure that once the programmer uses a wrong data, it can immediately return an error. This ensures that programmers can fix problems quickly.

## **2.3 Definition of Information Hiding**

Information hiding refers to that the specific information (process or data) contained in a module cannot be accessed by other modules that do not need such information when designing and determining the module[5]. That is to say, the specific implementation details in one module are not disclosed, and only some simple interfaces are reserved for other modules to access. Developers do not need to spend a lot of time to understand how to realize the function inside the module when writing the program. They will call the module to complete the corresponding function.

## **2.4 Benefits of Information Hiding**

Information hiding has some benefits. This paper discusses the following two aspects: hiding complexity and hiding change sources.

(1) Hidden complexity: a mature software is very complex, whether from a logical or code point of view. Any developer needs to spend a lot of time in understanding the code when he is perfecting the software function and solving the bug. In addition, a commercial software is often developed by a development team rather than individuals. In the face of complex functions and a wide range of code, how to effectively let everyone in the development team cooperate with each other is also a problem. Information hiding can effectively solve these two problems. Developers turn a complex system into different modules. Each module

implements its own specific functions, and there is no overlap between modules. Information hiding makes it unnecessary for each module to figure out how other modules realize the desired functions, just to know how to use this module, that is to say, it saves time to understand other modules when analyzing problems and understanding the code of this module. This can simplify a complex system. On the other hand, information hiding enables other people in the same development team to know what functions this module has implemented only through the class or function name. When the desired function is needed, the corresponding module is directly used without considering how this module is implemented.

(2) Hide the source of change: there is a problem in software development, that is, the requirements of software are changing with the development progress. When a software development is gradually completed, the change of requirements means that it takes a lot of time to modify the previously completed functions. This problem often leads to continuous rework and modification of the development work and continuous extension of the development time. Through information hiding, developers hide different functions into different classes. When developers need to modify a function, they can directly find the corresponding class to modify. This can not only reduce the number of code to be modified, but also ensure that other functions do not change

when modifying this function.

## References

- [1] ALTELS, DAVID. TEST-DRIVEN DEVELOPMENT - A PRACTICAL GUIDE[M]// Test Driven development: A Practical Guide. Prentice Hall Professional Technical Reference, 2003.
- [2] Zhang Yong. Test Driven Development [J]. Consumer guide , 2007(11):222-222.
- [3] Rose S , Wynne M , Hellesoy A . The Cucumber for Java Book: Behaviour-Driven Development for Testers and Developers[M]// The Cucumber Book: Behaviour-Driven Development for Testers and Developers. Pragmatic Bookshelf, 2012.
- [4] Liu Haiyan, Zheng Xiangyu, Chen Kejin, et al. Application Research of Feature Driven Development [J]. Computer Engineering, 2005, 31(21):66-68.
- [5] Xiong Baixiang. Information hiding in software design [J]. Enterprise technology development (the second half of the month) , 2011, 29(10):81-81.

### FINAL GRADE

70/100

### GENERAL COMMENTS

#### Instructor

Task 1 report (35%). 22

Task 1 use of git (15%) 13

Task 2 report (20%) 11

Task 2 code implementation (30%) 24

#### Comments

=====

BDD and TDD discussed, but a little more detail could have been provided including some examples of how to use them.

Cohesion and coupling are discussed, although some parts were a little less readable than others.

Implementation does contain some errors e.g. you can place multiple time-bombs.

Good usage of Git.

---

PAGE 4

---

PAGE 5

---

PAGE 6

---

PAGE 7

---

PAGE 8

---

PAGE 9

---

PAGE 10

---

PAGE 11

---

PAGE 12

---



### Comment 1

Sort of??

---

PAGE 13

---

PAGE 14

---

PAGE 15

---

PAGE 16

---

PAGE 17

---

PAGE 18

---

PAGE 19

---