Quiz 04: Testing

Due Feb 22 at 11pm **Points** 50 **Questions** 11 **Available** until May 15 at 11:59pm
Time Limit None

Instructions

You **may** use the slides from the lecture and other sources to answer these questions. Please be sure to cite any references but be sure to answer the following questions in your own words. Do NOT simply cut and paste the information from the slides. You will receive a score of 0 if you copy the prose from the slides.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	4,744 minutes	47.5 out of 50

Score for this quiz: **47.5** out of 50
Submitted Feb 22 at 1:32am
This attempt took 4,744 minutes.

Question 1 — 5 / 5 pts

Why should you only write a few tests at a time when practicing TDD?

Your Answer:

We should only write a few tests cases because firstly it is simple and easy task to complete and if the tests fail, we get feedback, and it is easy to debug since it is a few lines of code. Whereas writing more test cases will take more time and if the test cases fail it will be difficult to find a bug and correct it, hence might loose our motivation to work. Typical cycle of a test case is that a developer writes some test and then runs them, most of them fail then he writes some code to fix them and then runs them again, some of them fail again. This process goes on until all the test cases are passes. This process is easy and takes less time with few test cases and an excess amount of time and is more tedious and harder with a lot of test cases.

Since tests should be written to test small pieces of code, the more tests that are written the more code would have to be written at once, which goes against good practices. By writing a few tests at a time, you are checking small portions at a time which ensures better test coverage.

Question 2

5 / 5 pts

Describe two challenges associated of agile testing compared to waterfall.

Your Answer:

The two Challenges associated are

Firstly, the tests are probably lacking because of continuous integration. Continuous changes to features make testing difficult, plus the Schedule for development and testing is limited. Therefore,

changes are made so testing of new changes might be missed by the developer

Secondly, as there are constant changes in the code after each iteration. Changes happen so fast that some problems may be overlooked. Therefore, the possibility of code break existing feature becomes higher.

Changes happen quickly and problems may be overlooked

Schedules for development and test are highly compressed

Frequent changes to features makes testing challenging

Question 3

5 / 5 pts

Compare and contrast testing with the waterfall and agile methods.

Your Answer:

Firstly, in Agile method unit testing, acceptance testing and regression testing is done after each iteration as we build the project. Whereas in waterfall method unit testing, acceptance testing and regression testing is done only after the project has been created.

Secondly, in Agile method testing team and development team collaborate. Therefore, there is no time delay between coding and testing. Whereas in Waterfall method testing team and development team don’t collaborate. Hence, there is a normal time delay between coding and testing.

Thirdly, in agile method different testing levels overlap. Whereas in waterfall method testing levels can’t overlap.

Question 4

5 / 5 pts

What types of feedback do tests provide to programmers?

What types of feedback do tests provide to customers?

Your Answer:

Programmers gets feedback when their tests pass and when their test cases fail. This helps programmers to identify the issue in the code and make changes accordingly.

For programmers - location of bugs, number of passing/failing tests, what areas need improvement

Customer gets feedback when the test case pass or fail and get insights about how developers are doing and performing on user stories.

Customers - how developers are doing, what is working/what isn’t in the software

For programmers - location of bugs, number of passing/failing tests, what areas need improvement

Customers - how developers are doing, what is working/what isn’t in the software

Question 5

2.5 / 5 pts

Write a user story for some aspect of a party planning system using the standard BDD template proposed by Dan North.

Your Answer:

A party planning System

Narrative:

As a party organizer

I want to build a software for party planning.

So that users can easily arrange and make list of all the things required just in few seconds using the software.

Scenario 1:

The user will choose the guest list required and are to be invited for the party.

User will shortlist all the required products and food that will be necessary to arrange the party.

The venue and date are to be selected and finalized and shared to all the members of the guest list.

Once everything is finalized and approved the user can release anytime, he wants to release the details about the party.

Scenario 2:

The user has not decided about the venue, date, and time.

But the user has prepared the guestlist and notified about the party without a proper venue and time.

All the required products and food that will be necessary to arrange the party are already bought and available.

But the user cannot proceed without having a finalized venue and time.

Scenario 3:

The user has decided the venue and started the party planning.

The user prepares the guestlist and invites them all.

The user shortlists all the required products and food that will be necessary to arrange the party, but he fails to arrange all the requirements he shortlisted for the party.

Due to non-availability or failure he fails to arrange the product and food for the party.

In this case, the party will fail and should be handled asap by the user

Given some initial context (the givens), When an event occurs, Then ensure some outcomes.

Follow the template. slide 47. Given some initial context (the givens), When an event occurs, Then ensure some outcomes.

Question 6

5 / 5 pts

What is the purpose of the fixture in FitNesse?

Your Answer:

The purpose of fixture in FitNesse is to establish a connection between testing system and application. When a "Test" is commenced, fixture is called to process table.

The Fixture assigns the main application code and maps the user test to relevant code. It is like xUnit testing it extends base class and may create a base object to test multiple cases.

It acts as the connection between test system and application. It delegates to underlying application code and maps the user’s test to the relevant application code.

It acts as the connection between test system and application. It delegates to underlying application code and maps the user’s test to the relevant application code.

Question 7

5 / 5 pts

What is the difference between unit testing and acceptance testing?

Your Answer:

Unit testing is testing of small code and to find out if there is any error or not. It is good but assert that the code works as expected. But this testing may not always check the Stakeholder scenarios. Whereas in Acceptance testing is based on scenario provided by the stakeholder and usually each user stories will have at least one associated acceptance Test. They work as black box testing.

Unit testing ensures that recently changed units or features work correctly in the code. Whereas acceptance testing is where the customer verifies that the system performs as expected and meets requirements.

Unit testing ensures that recently changed units or features work correctly in the code, acceptance testing is where the customer verifies that the system performs as expected and meets requirements.

Question 8

5 / 5 pts

How is testing different from debugging?

Your Answer:

Testing means to find bugs and errors in the code. Whereas Debugging means to fix the bugs found during testing process. Testing is done by tester whereas debugging is done by either programmer or developer. In testing tester does not need to know the background of the project whereas in debugging developer cannot fix the bug without the knowledge of the project. Testing can either be manual or automated whereas debugging can never be automated it is always manual. There are

different types of testing whereas debugging doesn't have any types. Testing can only be done after the code is completed whereas debugging is done after execution of test cases.

Testing is the process where you identify and locate errors in the code. This does not necessarily change or do anything to the code itself. Debugging is when you actually go through the program and make the changes necessary in order for tests to pass.

Question 9

5 / 5 pts

Describe one strategy for prioritizing customer reported bugs when using Scrum?

Your Answer:

When we ask a customer about their feelings about a certain new feature which is added after an iteration or the bugs they report. Because in Agile customer feedback is very important. One way to prioritize customer reported bugs or feedback is to add them to the product backlog as a new user story. This provides an easy way to keep track of them. Once they are in the backlog, the Product Owner can then prioritize the bug user story by evaluating the importance of a bug fix or feature request is by looking at the value our implementation is causing and finally decide when to fix that particular bug. So that it could be added to the next sprint.

One way to prioritize customer reported bugs when using Scrum is to add them to the product backlog as a new user story. This provides an easy way to keep track of them, and depending on the severity, decide when they will be fixed. Once they are in the backlog, the Product Owner can then prioritize the bug user story so that it could be added to the next sprint

Question 10

5 / 5 pts

What is the advantage of frequent integration?

Your Answer:

The Advantages of frequent integration are firstly it allows changes in smaller code one at a time and then integrate these small pieces of code because if there are large sections of code that has not been tested and a test fails, the error could be anywhere within that entire code. Since the code is small identifying bugs becomes similar and we can easily debug them. The more frequently you integrate, the more accurately. The release of a features to the end user is quicker. Frequent integration also helps in Customer satisfaction.

One advantage to using frequent integration is that it make it much easier to find and locate errors in programs. The more frequently you integrate, the more accurately you will be able to find errors that arise. If there are large sections of code that has not been tested and a test fails, the error could be anywhere within that entire code. With frequent integration, there is small portion of code the error could be in, and therefore it is easier to fix.

Question 11

0 / 0 pts

"I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination. I further pledge that I have not copied any material from a book, article, the Internet or any other source except where I have expressly cited the source."

Correct!

- (●) True
- ( ) False

Quiz Score: **47.5** out of 50