*Nova Southeastern University*

*College of Engineering and Computing*

*CISC 680 - Software Engineering - CRN – 21741*

*Term Code: Fall 2019 (202020) Course*

*Dates: 08/19/2019 - 12/08/2019 – On Line*

**Question 1:**

This is a software Testability Checklist

1. Operability – the better it works the more efficiently it can be tested

2. Observability – what you see is what you test

3. Controllability – the better software can be controlled the more testing can be automated and optimized

4. Decomposability – by controlling the scope of testing, the more quickly problems can be isolated and retested intelligently

5. Simplicity – the less there is to test, the more quickly we can test

6. Stability – the fewer the changes, the fewer the disruptions to testing

7. Understandability – the more information known, the smarter the testing

Choose two and explain why they have higher merit than the others, remember to provide enough depth so that you convey that you understand the depth of the question.

From the software testability checklist Understandability has the highest merit. In software testing this is important because the more a tester understands about the project, the better they can go through and ensure the proper requirements have been fulfilled. Since testing is an integral part of software development, understanding the code and its requirements should weigh pretty high on that scale. It makes little sense to have a tester test something that they do not understand or comprehend. Ultimately, having more insight and knowledge leads to higher quality tests. Software development is all about user satisfaction and it is up to the tester to verify that the product meets the client’s demands. Having a tester who truly understands the needs of the clients and the programs technical ability, leads to better testing and happier clients.

Second on that list would be Observability, this is the concept of the tester only being allow to test what they can physically see. This is done to mimic testing from a user’s point of view. This is important because it helps developers ensure that all the clients’ requests can be observed and accounted for. The more observable information is in the program, the deeper a tester can go into finding issues and bugs. If something is being tested but the outcome isn’t easily observed then it becomes an issue from a testing standpoint. It is recommended requirements be observable so they can tested and fixed before the final product is revealed.

**Question 2:**

In OO Testing,

What is the difference between surface and deep structure testing and how does it benefit the overall testing of OO designs?

When should they be used when why is the environment important?

What kind of project would require each, give three examples of each?

Object-Oriented testing allows testers and software engineers to test classes, objects, functions, and over all code structure in an OO architecture. This allows for a more robust understanding of tests performed in the OOP application.

Surface structure testing allows testers to test the application from the user’s point of view and typically operates more on the front end of the application. Surface structure testing involves manipulating objects and performing tasks exhausting all user choices. In order to do this testers need to understand the process in which the user goes through to complete tasks. Testers will start at the surface of these tasks and then exhaust resources inwards as the process begins to pan out. Essentially surface structure testing’s is testing performed from the top down in the point of view of the user and then trying all combinations possible. This is beneficial to the overall OO testing because it allows testers to better understand requirements from a user’s point of view. Surface structure testing would be used in projects that weigh heavy importance on user interfaces.

Some examples would be a drag and drop map application with a nice graphical interface. Another could be a game with a main character walking a storyboard. Lastly, another example of a program you would test surface structure testing would be some kind of banking application, where surface structure would be tested from initial log in, all possible transactions, then logout.

Deep structure testing can be thought of opposite of surface structure testing in regards to it is testing done from more of a inwards out approach behind the scenes from the user, while typically performed on the back end. This structure can only be understood by reviewing the underlying code. Deep structure testing allows for testing of behaviors, dependencies, and communications between objects and classes. This benefits OO testing because it allows testers to better understand back end process and relationship that are not easily observable to the client or on the front end. A tester would implement deep structure testing when relations ships between objects or process are to be better understood and tested.

An example of this would be if you have a program with an user object and an admin object. Deep structure testing would test the relationship between the two and see how much functionality a user would have compared to an admin.Another example would be a dog class and a cat class that both implement the same animal interface. Deep structure testing would test how both dog and cat implement the animal class differently. Last another example of deep structure testing would be to see how all inherited classes interact with each other. This could be conveyed as a child class inheriting a parent class inheriting a grandparent class and how they affect each other is tested in deep structure testing.

**Question 3:**

In Formal modeling and verification what is the Z Specification language?

Why is it used and what are the drawbacks of formal modeling, Give three examples of software projects (applications) that would require Formal modeling.

The Z specification language is one of the most widely used and regarded for formal modeling and verification. This language allow for modularity through schemas and allows processes to be visualized and later built. It is used to design and map out processes to later be built regardless of what programming language is chosen. This allows for a single uniform design to be adopted and then applied later on down the line. These graphical implementations is used among colleagues to verify business processes and set a baseline design. Some draw backs of formal modeling is that it is pretty technical in the fact that it is not common knowledge for most people. This can be time consuming and expensive to an organization to have to pay for someone with the technical prowess or to have someone trained in formal modeling. One example of an application would use formal modeling would be game theory where complex algorithmic process would need to be proved out in a formal specification language. Another example would be in GPS imaging, where complex processes would be need to be mapped out regarding the multiple data entry points coming into the system. Lastly, another example of a program that would use formal specification would be any project regarding science or physics such that the complex equations would need to be proved out before being implemented in a native programming language.

**-**

**Question 4:**

In Measurement Process Activities, the following list applies:

1. *Formulation* – derivation of software measures and metrics appropriate for software representation being considered

2. *Collection* – mechanism used to accumulate the date used to derive the software metrics

3. *Analysis* – computation of metrics

4. *Interpretation* – evaluation of metrics that results in gaining insight into quality of the work product

Of the above four items, list the order of focus,, highest to lower which items requires more work than others, Give a detailed explanation of why you think this is so.

**Question 5:**

What are the major differences between Process and Project Metrics and why are they important to software engineering?

**Question 6:**

How does Software Re-engineering fit into the agile development cycle, explain in detail, how to combine re-engineering process model and the agile process model

**Question 7:**

If you are in an agile development cycle how would you incorporate the use of UML, does it have a place and how strongly should it be implemented?

**Question 8:**

Regarding Clear-Box, State-Box and Black-Box quality management approaches,

Explain the uses of each?

Give an example of each?

Why you would use 1 over the other 2 in each case?