**UML Design Modeling**

Travis Henagan

CST499: Capstone for Computer Software Technology

Professor Amr Elchouemi

July 12, 2021

**UML Design Modeling**

Testing is an important and mandatory part of the software development life cycle (SDLC). There are different ways that are included in the testing of the software. During the SDLC, different types of testing is done to ensure that the development is flowing well, and errors can be detected early for correction. There are different levels of testing that take place at different levels of the SDLC. In this paper I will discuss in detail the different levels of testing which includes component testing, integration testing, system testing, and acceptance testing.

The component test level systematically tests the software units for the first time. Each software unit (component) is tested individually based on component requirements, and the component design (Spillner et al., 2014). The tests objects used for component tests are broken down and isolated to prevent external influences. The test objects that are used for component tests are done by developers and the test are done involving developers. The component tests are done with multiple test cases, in which each test case has a partial functionality. Component test usually identify minor mistakes done by developers such as incorrect calculations. Component testing usually includes access to the source code, which allows for us to use white box testing. In testing today, the testing is done with test first programming. Test first programming is used to design the and automate the tests first and program the components afterwards (Spillner et al., 2014).

The integration test level takes components from the component test and brings them together to enhance the testing and to see if the components collaborate. The main goal of integration test is to uncover flaws in the interfaces and the interaction of the components that are integrated. The test objects that are used for integration test can be internal interfaces between components and consists of configuration programs and configuration data (Spillner et al., 2014). During integration test, test drivers are used to send test data to the test objects and the outcomes are recorded. When the test objects are tried, they will disclose flaws between components that are being tested together. There are different strategies that are used to perform integration test, an efficient strategy is top-down integration.  “Top-down testing is done by integrating or joining two or more modules by moving down from top to bottom through control flow of architecture structure. In these, high-level modules are tested first, and then low-level modules are tested. Then, finally, integration is done to ensure that system is working properly” ("Steps in Top-Down Integration Testing - GeeksforGeeks", 2020). This allows for the top-level components to be the test driver throughout the integration test. Integration is a kind of black-box testing.

The system test level is done after integration test in which it detects failures to make sure it corresponds with the specifications and requirements. The references material that may be used for system testing can be requirements and specifications of the software system, use cases, risk analysis, applicable norms, and standards. The main goal of the system test is to make sure all components are properly integrated and accessible (Homes, 2012) and to detect any irregularity between the units that are integrated together. It is important to point out that when system tests are being done, the software is complete and should be as close to what the user will use as possible. At this point system hardware, software, device drivers etc., should be installed on the tests platform for completeness. Different types of system test include performance testing, load testing, stress testing and scalability testing. The quality of the data becomes more important when at the system test level. The tests confirm that the entire system meets the required functional and nonfunctional requirements (Spillner et al., 2014). Systematic testing of the models can detect active faults prior to implementation, which can decrease the time it will take to correct the errors later in developmental phases (Kawane, 2003).

The acceptance test level is the tests of “the complete software, its documentation (user manuals, installation and maintenance documentation, etc.), all necessary configuration items, forms, reports and statistics from previous test levels, user processes” (Homes, 2012, p. 64). As the name implies, acceptance tests are done to verify the acceptance from the customer or user of the software. The product must be fine-tuned and ready for customer approval. Different types of acceptance test include user acceptance testing, and business acceptance testing. Each type of acceptance testing explores if the software working properly for the user and if the products align with the business goals. The acceptance tests are usually done with the help of the customer to confirm the software product.

As different test levels are reached, UML diagrams are used to do the testing. I created three UML diagrams for a system that I am developing for student college course registration. The images of each UML document are below which includes a class diagram (figure 1), sequence diagram (figure 2), and a use case diagram (figure 3).

**Figure 1**

**Diagram

Description automatically generated**

**Figure 2**

**Diagram

Description automatically generated**

**Figure 3**

**Diagram

Description automatically generated**

References

Homès, B. (2012). Fundamentals of software testing. Wiley.

Kawane, N. (2003, November). Fault detection effectiveness of UML design model test adequacy criteria. In *Supplementary Proceedings of the International Symposium on Software Reliability Engineering* (pp. 327-328).

*Steps in Top Down Integration Testing - GeeksforGeeks*. GeeksforGeeks. (2020). Retrieved from https://www.geeksforgeeks.org/steps-in-top-down-integration-testing/.

Spillner, A., Linz, T., & Schaefer, H. (2014). Software testing foundations: A study guide for the certified tester exam (4th ed.). Rocky Nook.