**UML Design Modeling**

Hayden Herrig

CST 499: Capstone for Computer Software Technology

Prof. Charmelia Butler

February 12, 2024

**UML Design and Testing**

Testing is an important process in the software development process. Being able to verify that all the pieces of the software system are functioning as intended is important to delivering a quality software system that meets all the requirements. Performing tests also helps to detect errors earlier in the development process. Catching errors early helps to reduce costs, reduce development time, and improve maintainability. It is important to make sure that the tests being conducted are specified. If tests are not specified, then testers do not know what the system is meant to accomplish. UML diagrams can help identify what the system should be tested for. UML diagrams provide a visual for the intended operation of the system at many different levels. They can provide insight on both static and dynamic aspects of the system. This helps design tests at all levels of testing. There are four levels of testing which consists of component, integration, system, and acceptance testing. Each of these levels of testing helps to ensure that the system functions as intended and meets the previously specified requirements.

**UML Diagrams**

A diagram of a person's structure

Description automatically generated

**(FIG 1. Use Case Diagram)**

A diagram of a course

Description automatically generated

**(FIG 2. Class Diagram)**

A diagram of a student registration portal

Description automatically generated

**(FIG 3. Sequence Diagram)**

A diagram of a student registration

Description automatically generated

**(FIG 4. Activity Diagram)**

**Levels of Testing**

Component Testing will ensure that all of the individual objects that are developed for the system will function as intended. This level of testing will ensure that the components of the system such as scripts and other software components have been thoroughly tested (Spillner et al., 2014). Component testing will include testing all of the objects identified in FIG 2. Thoroughly testing these objects will ensure the individual objects are not the cause of errors later on in the testing process. For the enrollment system the components of the system consists of the student, course, catalog, and schedule. The student component will be tested by making sure students can register an account, then log into the account. It will also be tested to ensure they can access their schedule and the course catalog. The course component will make sure the course and its information are available to be displayed and students are able to enroll in the course. It will also be tested to ensure that when a student enrolls in or drops a class the number of students will update to display an accurate enrollment number. The catalog component of the system will be tested to ensure that all of the available courses are displayed to the students looking to enroll in the course. It will be tested to ensure students are able to easily browse all courses available to the student. Lastly, the schedule component will be tested to ensure that it accurately displays courses the student is enrolled in and that students can drop courses if necessary. If a student enrolls in a new course the schedule should be updated to reflect this as well as if a student drops a course the schedule should reflect this action.

Integration testing is performed to ensure that when the units are linked together the combined units will function as intended. The aim of integration testing is to test the interfaces between the modules and expose any defects that may appear when the components are integrated and need to interact with each other (Awati, 2022). This testing will ensure that any components mentioned above will be able to communicate with any other necessary components. For this testing the components to be linked will be linked together and made sure that they communicate appropriately based on the previously provided requirements. To determine which components, need to be linked and tested during integration testing refer to FIG 2.

System testing is meant to check that the integrated product meets the specified functional and non-functional requirements (GeeksforGeeks, 2023). The system tests are used to run the entirety of the system in a setting as close to the intended system environment. System tests are also meant to verify system and user documentation such as manuals and training materials. This testing will ensure that the system as a whole will meet the requirements provided by the client. Once integration testing is completed all components will be integrated to build the final system and ensure that the system functions as intended. For reference on the intended functionality of the system refer to figures 1, 3, and 4.

Acceptance testing is performed in a user environment that resembles the production environment. User acceptance testing verifies that the delivered system meets the user’s requirements and is ready to be used in the real world (GeeksforGeeks, 2023). For this testing the final product contract acceptance testing. This allows the system to be tested to the point to where it meets all requirements requested by the client. Acceptance testing will be conducted on all necessary platforms to ensure successful operation across all necessary systems and devices.

**Conclusion**

Testing is an important aspect of developing software systems. It ensures that a quality software system is delivered, and that the system meets all the previously specified requirements. Testing at all levels helps to ensure all individual components function as intended, modules communicate as intended, the system functions as intended, and that it meets all requirements. Utilizing UML diagrams to understand the system, its components, and how they interact with each other helps to develop tests that will ensure the system operates as the diagrams depict. Utilizing UML diagrams to help design tests at all levels ensures thorough testing of the system.

# References:

Awati, R. (2022). integration testing or integration and testing (I&T). Software Quality. <https://www.techtarget.com/searchsoftwarequality/definition/integration-testing>

GeeksforGeeks. (2023). Software Engineering White box Testing. GeeksforGeeks. https://www.geeksforgeeks.org/software-engineering-white-box-testing/

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