**UML Design Modeling**

Jonathan Schaub

The University of Arizona Global Campus

CST 499 Capstone for Computer Software Technology

Dr. Charmelia Butler

October 23, 2023

**Figure 1**

*Use Case Diagram*

A diagram of a course registration

Description automatically generated

**Figure 2**

*Class Diagram*

A computer screen shot of a computer

Description automatically generated

**Figure 3**

*Sequence Diagram*

A screen shot of a computer

Description automatically generated

**Figure 4**

*State Diagram*

A screenshot of a computer

Description automatically generated

**Introduction**

Software development is an ever-evolving landscape where software quality is the key to creating a software product that is high-quality and extends the lifecycle of the software. Software testing is a multi-facet process that examines and tests various levels of software from software components to how they interact, and the user’s experience, to ensure they meet or exceed specified software requirements. Software testing acts as a sort of quality-control by digging into a software application from the ground up to test individual components, the components when interacting with each other, the system as a whole, and the end-user’s experience. This paper will examine the layers of software testing from the most granular component testing up to acceptance testing of the fully assembled product by end-users.

**Component Testing**

Components are the foundation of a software and component testing, or unit testing, is the ground layer that can determine software quality. Software components or units are the building blocks of all software, and component-level testing hones in on these building blocks in order to confirm the software is foundationally sound. According to Tsui, et al., (2018) unit testing examines each of the software components individually in order to verify each unit adequately serves its purpose. Component-level testing relies on the principle that high-quality software can only be achieved when the foundation is structurally sound. According to Dhore et al., (2023) unit testing is the first line of defense to prevent software errors and anomalies. Component testing also serves as an early detector of bugs and can ensure a software components integrity (Dhore, et al., 2023).

**Integration Testing**

Integration testing is a means of ensuring cohesiveness between the various software components. As the various software component “building blocks” are assembled, they must link and interact with each other in a way that is both cohesive and efficient. Integration testing is a means of ensuring shifting gears from individual component functionality to gauging how each component interacts with one another and to identify components that may seem to work well individually but may work abnormally when combined. The goal of integration testing is assessing the quality and reliability of the units when they are interfaced together (Tsui, et al., 2018). A key part of integration testing is to validate data its flow data between integrated components (Dhore, et al., 2023). Another aspect of integration testing is to test the inter-component interactions and dependencies to ensure they are cohesive for the overall functionality of the software (Pillai, et al., 2019). At its most basic level, integration testing “ensures usability, stability, and reliability of the units (Dhore, et al., 2023).

**System Testing**

A step above integration testing is system testing, which offers a holistic approach to software testing that evaluates the software in its entirety. According to Dhore (et al., 2023, p. 456), system testing “assesses he behavior of a machine and its compliance in a fully integrated consumer product with the software requirements.” Software testing can range from testing the overall load management, security, and performance of a fully-assembled software application. Both black-box and white-box system testing are crucial in the software testing phase because they both examine the internal structure of the software and the functionality of the software as it is perceived by the end-user (Tsui, et al., 2018).

**Acceptance Testing**

After the system has been thoroughly vetted from a technical aspect, the final component of testing happens with the end-user and whether they can accept the software for use in their daily activities. According to the research of Pillai, et al., (2019. p. 3), users of the software assess it to confirm the software has a full grasp on the requirements that are mandatory for real-world application. Dhore (et al., 2023) also suggests that factors like stress tolerance, load management, security, and overall performance are key in the acceptance testing process. System testing must be thorough and complete to prescribed levels in order to ensure it is ready to go live by being deemed adequate for its purpose, reliable, and acceptable for the users.

**Conclusion**

Software across the globe thrives when it is high-quality, and this is underscored by the layers of software testing described in this paper. Software testing starts at the foundational level with component testing and proceeds through the integration of units through to the completed system and eventually to the end-users of the software. Everything from the foundational strength provided by component testing or efficiency due to the harmony provided by integration testing, testing bolsters quality to ensure the finished, deployed product is complete, usable, secure, and will function adequately for years to come as long as it maintained properly. Both Dhore, et al., (2023), and Tsui, et al., (2018) state that the intricacies of software testing are more than just boxes to check when developing software, and are rather means of ensuring a software product meets the high standards of current and potentially future users. In an ever-evolving landscape of software development, delivering high-quality software products is key, and it can only be achieved through rigorous and complete software testing at all levels.

**References**

Dhore, P., Wadhwa, L., Shinde, P., Chaudhri, D., & Vyas, P. (2023). Brief Review on Different Manual Software Testing Approaches & Procedure. *Journal of Pharmaceutical Negative Results*, *14*, 455–464. <https://doi.org/10.47750/pnr.2023.14.S02.56>

Pillai, N. S. R., Hemamalini, R. R., Padmavathy, V., & S., N. (2019). Framework for Multiple User Acceptance Testing to Avoid Chaos. *2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN), System, Computation, Automation and Networking (ICSCAN), 2019 IEEE International Conference On*, 1–6. <https://doi.org/10.1109/ICSCAN.2019.8878803>

Tsui, F., Karam, O., & Bernal, B. (2018). [*Essentials of software engineering*](https://uagc.instructure.com/courses/122857/modules/items/6249130)(4th ed.). Jones & Bartlett Learning.