**Week 5 – Testing Levels and UML Design Modeling**

Ben Merritt

The University of Arizona Global Campus

CST 499: Capstone for Computer Software Technology

Instructor: Dr. Charmelia Butler

December 11, 2023

**Week 5 – Testing Levels and UML Design Modeling**

In software development, ensuring a system's reliability, functionality, and performance is paramount. This necessitates a structured approach to testing that spans various levels of granularity and abstraction. Levels of testing provide a systematic and incremental method for scrutinizing software artifacts, ensuring that each component, interaction, and system meets specified criteria. This paper delves into key levels of testing, namely Component Testing, Integration Testing, System Testing, and Acceptance Testing, elucidating their roles in verifying different aspects of software artifacts. As we explore each level, we uncover how they contribute to the early detection and rectification of defects, ultimately culminating in a robust and user-approved software product (Tsui et al., 2018). These stages or phases of testing involve testing the software artifacts at different levels of granularity, integration, and abstraction. They can help test the software artifacts systematically and incrementally (Spillner et al., 2014). They can also help test the software artifacts for different objectives and criteria. Some examples of levels of testing are:

Component Testing: This is a level of testing that involves testing minor units or components of the software artifacts, such as classes, methods, functions, etc. It can help independently verify each unit or component's functionality and performance. It can also help find and fix defects early in development or testing (Spillner et al., 2014).

Integration Testing: This level of testing involves testing the interactions and interfaces between two or more units or components of the software artifacts. Integration testing can help you verify the functionality and performance of each unit or component in combination with others. Integration testing can also help find and fix defects at an intermediate stage of development or testing (Spillner et al., 2014).

System Testing: This level of testing involves testing the whole system or application as a single entity. System testing can help verify the functionality, performance, security, and compatibility of the system or application. System testing can also help find and fix defects later in development or testing (Spillner et al., 2014). System Testing tests if the system meets the specified requirements and is suitable for delivery to the end-users. For example, System Testing can verify that a software application for automotive smart power switches can interact with other software/hardware systems, such as sensors, actuators, controllers, etc., and perform the desired functions under different scenarios (Bala et al., 2023).

Acceptance Testing: This level of testing involves testing the system or application by the users or stakeholders to determine whether it meets their expectations and requirements. Acceptance testing can help verify the usability, reliability, and suitability of the system or application for its intended purpose. Acceptance testing can also help gain feedback and approval from the users or stakeholders (Spillner et al., 2014).

In conclusion, the multi-tiered approach of levels of testing offers a comprehensive strategy for software quality assurance. Component Testing ensures the integrity of minor units, Integration Testing validates seamless interactions, System Testing evaluates holistic functionality, and Acceptance Testing aligns the software with user expectations (Tsui et al., 2018). By systematically progressing through these levels, developers can fortify their software against potential defects and deliver a product that functions optimally and satisfies the end-users, marking the pinnacle of successful software development.

**UML Design Diagrams for the Course Registration System**

**Figure 1**. Online Course Registration System Use Case Diagram.

**A diagram of a flowchart

Description automatically generated**

**Figure 2.** Online Course Registration System Flow Diagram.

**A diagram of a user account

Description automatically generated**

**Figure 3.** Online Course Registration System Deployment Diagram.

**A diagram of a application

Description automatically generated**

**Figure 4.** Online Course Registration System Activity Diagram.

**A diagram of a flowchart

Description automatically generated**

**Figure 5.** Online Course Registration System State Diagram.

**A diagram of a flowchart

Description automatically generated**

**Figure 6.** Online Course Registration System Sequence Diagrams.

A diagram of course registration system

Description automatically generated

**Figure 7.** Online Course Registration System Class Diagram.

A diagram of a user

Description automatically generated

References

Bala, D., Ulbing, A., & Pathak, S. (2023). Advantages of System Level Testing and Modelling for Automotive Smart Power Switches. *2023 International Conference on Electrical Drives and Power Electronics (EDPE), Electrical Drives and Power Electronics (EDPE), 2023 International Conference On*, 1–7. https://doi.org/10.1109/EDPE58625.2023.10274058

Spillner, A., Linz, T., & Schaefer, H. (2014). [*Software testing foundations: A study guide for the certified tester exam* (4th ed.)](https://uagc.instructure.com/courses/121431/modules/items/6179936). Rocky Nook.

Tsui, F., Karam, O., & Bernal, B. (2018). [*Essentials of software engineering*](https://uagc.instructure.com/courses/124945/modules/items/6353220)(4th ed.). Jones & Bartlett Learning.