UML Design Modeling

Jose Herrera Medina

CST 499: Capstone for Computer Software Technology

Professor Joseph Rangitsch

September 20, 2024

# Different Levels of Software Testing

Software testing is critical to the software development lifecycle, ensuring that applications meet specified requirements, function as expected, and are defect-free. The process is divided into several levels, each focusing on different aspects of the software. The primary levels include component testing, integration testing, system testing, and acceptance testing. Each of these levels serves a unique role in ensuring the quality and functionality of software.

# Component Testing

Component testing, also known as unit testing, focuses on verifying individual components or modules of the software in isolation. A component is a distinct software part, such as a function, class, or procedure. This testing level ensures that each component behaves as expected according to its design and requirements (Dustin, 2015). During component testing, developers test the minor code units, often using mock objects or stubs to simulate the behavior of dependent components. Since these tests are typically automated, they allow for quick feedback during the development phase and help detect bugs early in the process.

One key benefit of component testing is that it isolates defects within specific modules, making pinpointing and correcting errors easier. However, because components are tested in isolation, it does not guarantee they will interact correctly with others, highlighting the need for higher-level testing (Vasin, 2021).

# Integration Testing

Integration testing examines how multiple components work together as a group. After individual components are verified through component testing, integration testing ensures they interact as expected when combined. This testing focuses on the interfaces and communication between components (Myers et al., 2017).

Integration testing can be performed incrementally, where components are integrated and tested in stages, or in a "big bang" approach, where all components are integrated and tested simultaneously. Incremental integration testing is preferred because it allows defects to be detected earlier and more easily isolated (Vasin, 2021). For instance, in software with a layered architecture, integration tests focus on interactions between the presentation, business logic, and data access layers. The goal is to ensure that data flows correctly between the layers and that all integrated modules are cohesive.

# System Testing

System testing evaluates the software, its functionality, performance, security, and overall behavior in a real-world environment. It is conducted after integration testing and involves running the software in an environment that mimics the production environment as closely as possible. System testing is crucial because it ensures the software meets its specified requirements and behaves as expected under different conditions (Dustin, 2015).

Unlike integration testing, which focuses on module interactions, system testing looks at the software from an end-user perspective. It checks whether the application as a whole is functioning correctly. Types of system testing include performance, security, and usability testing, all of which ensure the system's readiness for deployment.

# Acceptance Testing

Acceptance testing is the final level of testing and is often performed by the end users or stakeholders to verify whether the software meets their needs and requirements. This level of testing ensures that the software is ready for release and can be used effectively in a real-world environment (Myers et al., 2017). Acceptance testing may involve various forms, including user acceptance testing (UAT), where users validate that the software meets their expectations, and contract acceptance testing, where the software is verified against contractual agreements.

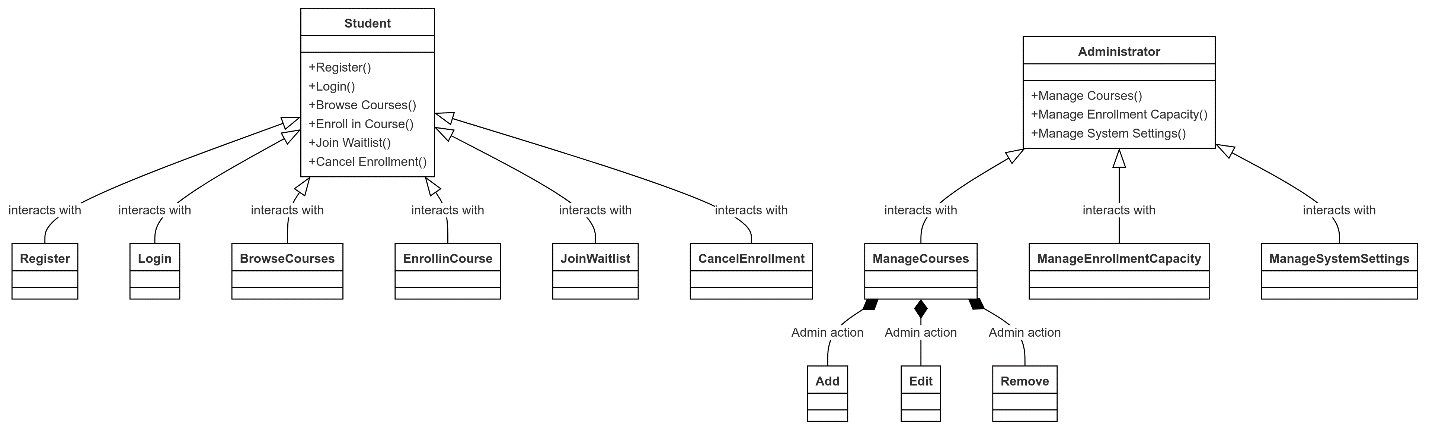
The primary objective of acceptance testing is to ensure that the software is ready for production deployment and provides value to the user. It often serves as the final checkpoint before the software is handed over for use in the operational environment. Successful acceptance testing results in the approval of the software for release, while failure can lead to further development or testing iterations.

# Conclusion

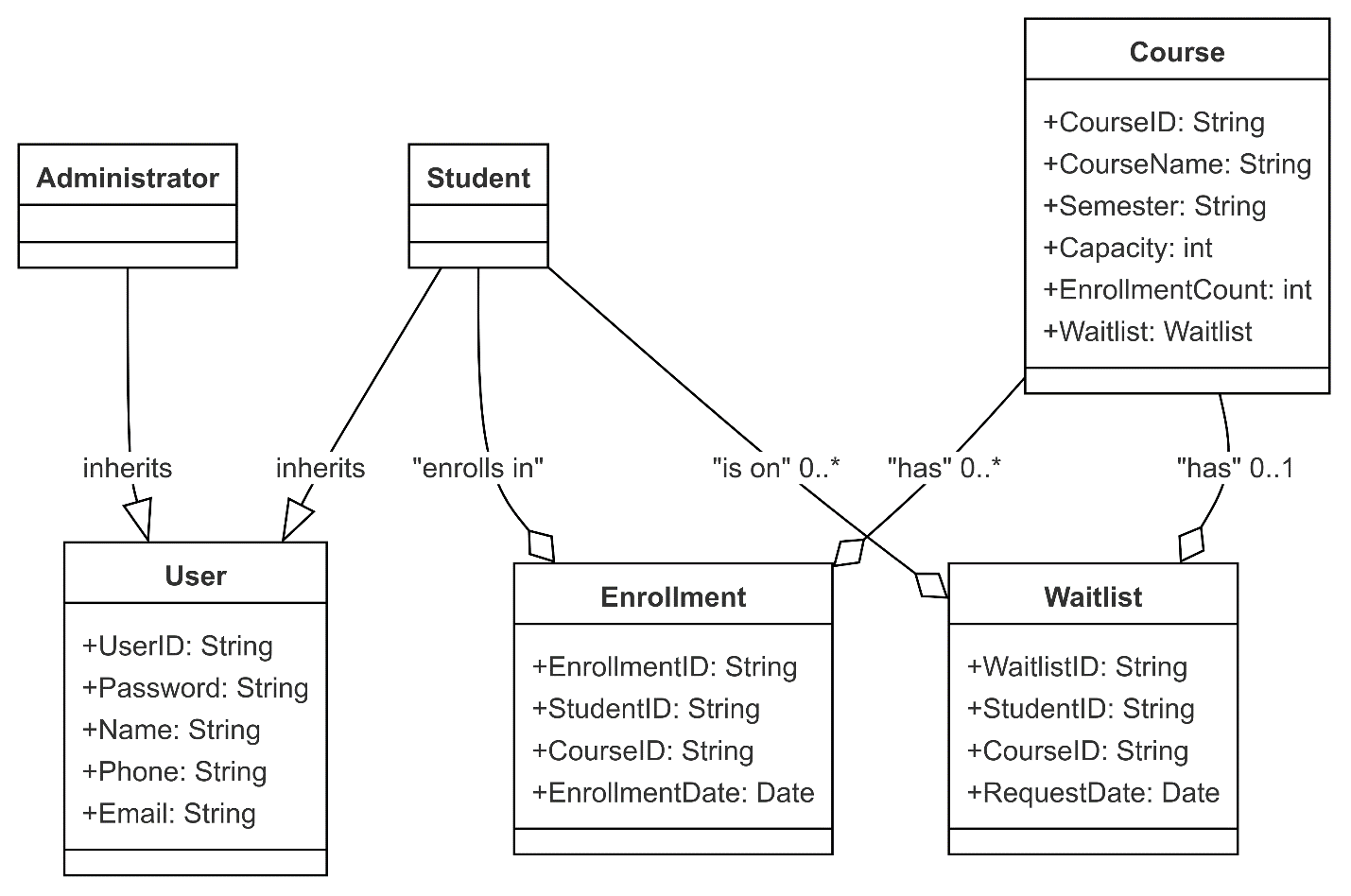
Each level of testing plays a crucial role in ensuring the quality and reliability of software. Component testing focuses on verifying individual modules, integration testing ensures components work together, and system testing evaluates the software's functionality. Acceptance testing confirms that the software meets user expectations. Together, these testing levels provide a comprehensive approach to identifying and resolving defects throughout development, ultimately contributing to higher-quality software.

# UML Diagrams

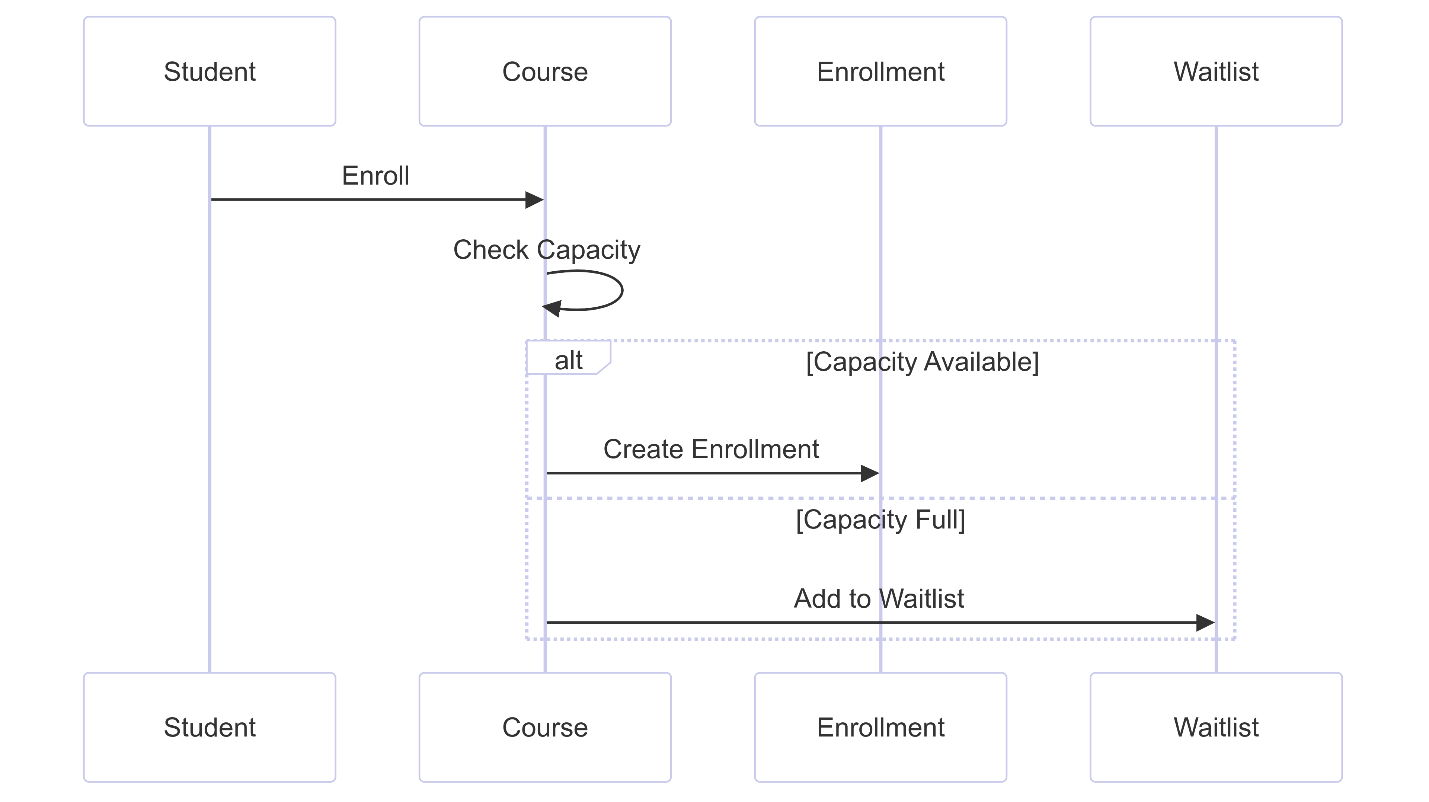
Case Diagram



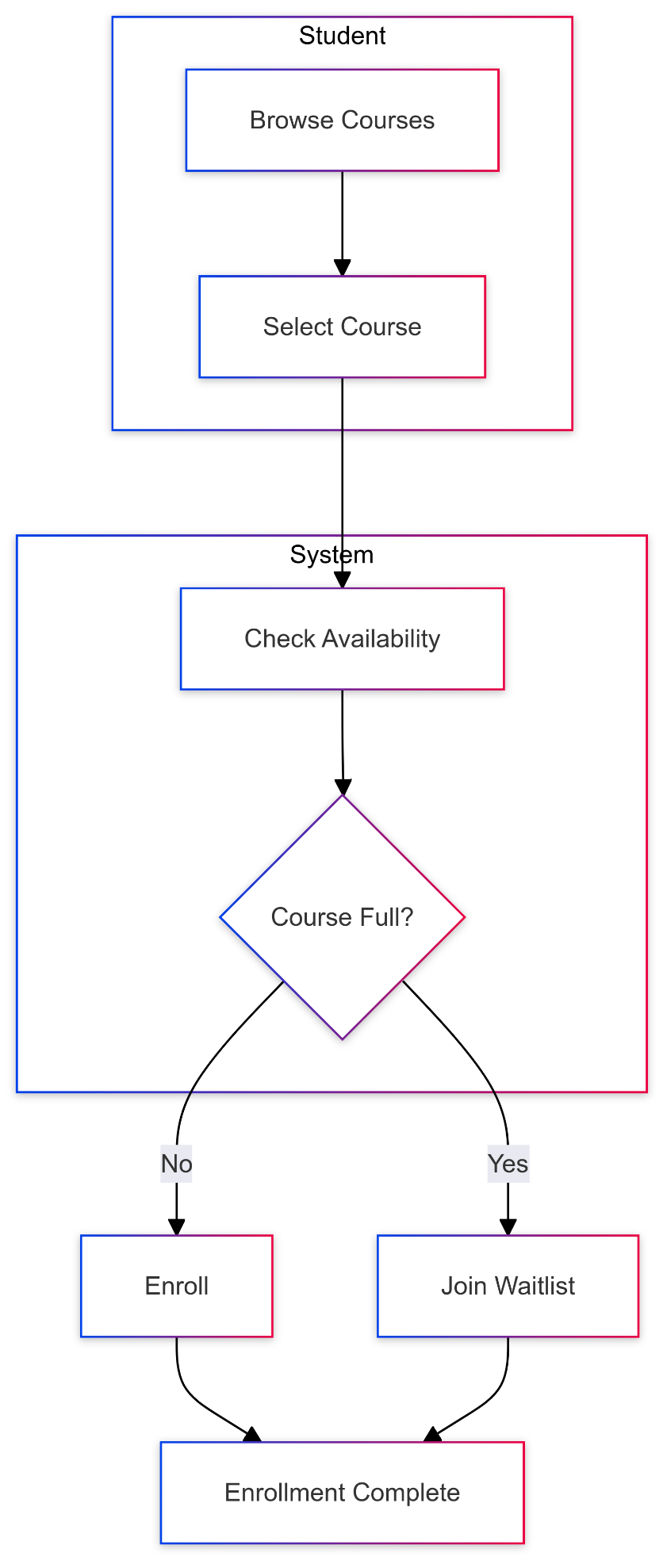
Class Diagram



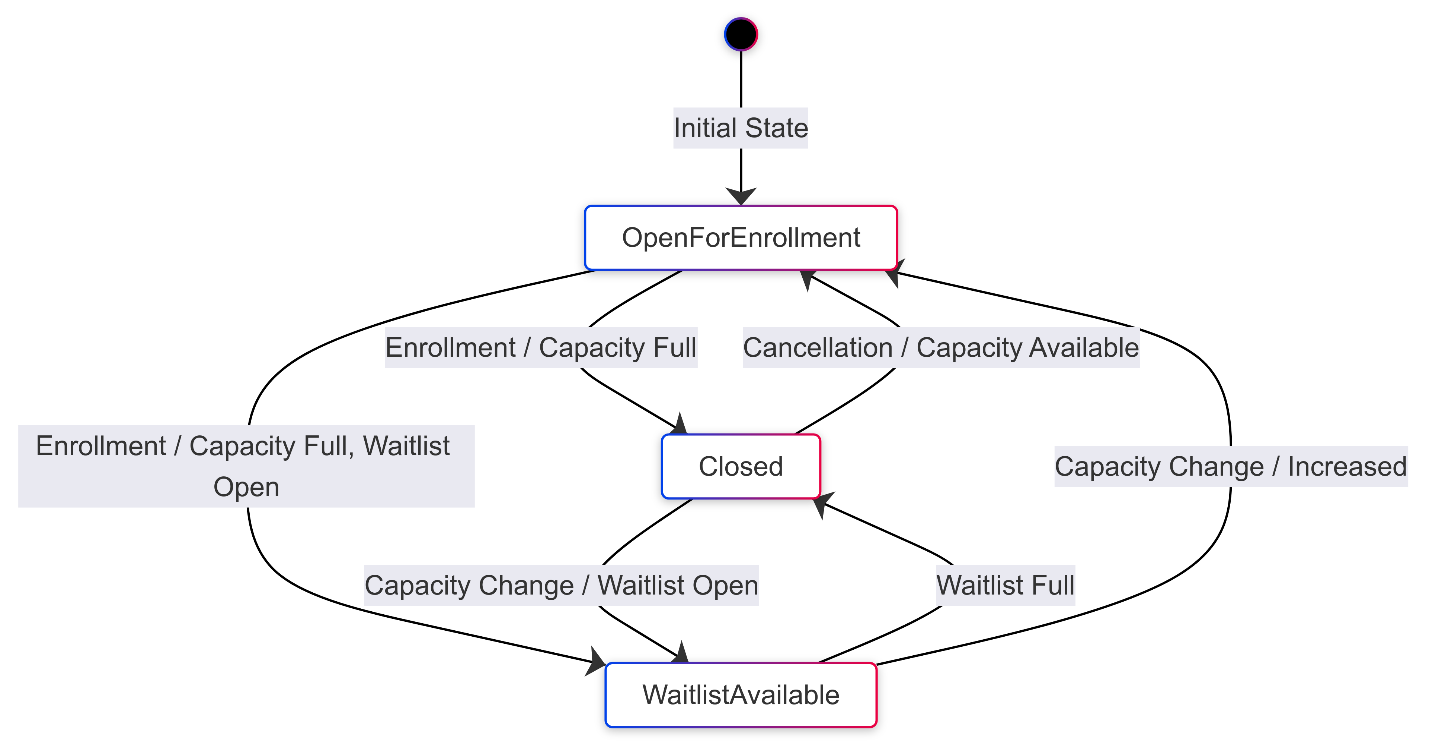
Sequence Diagram



Activity Diagram



State Diagram



References

Dustin, E. (2015). *Effective software testing: 50 specific ways to improve your testing*. Addison-Wesley Professional.

Myers, G. J., Sandler, C., & Badgett, T. (2017). *The art of software testing*. John Wiley & Sons.

Vasin, S. (2021). The importance of integration testing in modern software development. *Journal of Software Engineering Practices*, 12(3), 45–52.