**UML Design Modeling**

Benjamin Amrine

The University of Arizona Global Campus

SCT 499: Capstone for Computer Software Technology

Dr. Amr Elchouemi

October 10, 2022

**UML Design Modeling**

Time is money, and financial obligations and responsibilities will always be a critical factor when it comes to designing software. One way designers can reduce costs within any project is by implementing core solutions to common problems. Design patterns are exactly that, a generalized template for solving a particular problem that can help to reduce overhead and diminish the number of defects or errors that occur in all phases of the project. Testing any new product or system is imperative for producing a quality product. The following UML models and test levels will increase product quality, including functionality, reliability, usability, efficiency, maintainability, and portability (Homès, 2012). The associated UML models included within this paper are only some of the necessary documents needed for development and testing to begin but provide enough information for a broad overview of the procedures and structure of the system.

Each of the test levels listed below the UML model section is critical for reducing defects and providing the desired software quality, reliability, system assurance, optimum performance, a higher return on investment, transparency to management on progress, and the overall usability of the system. Within the requirements elicitation phase, the purpose and desired characteristics of the system are defined, producing the overall acceptance criteria for the end product. It is extremely valuable that testers are included within this phase of the development life cycle for them to understand the final phases and exit criteria of acceptance testing. During the functional system design, requirements are mapped to functions and provide the necessary information for system testing, including the desired interactions of all system components and "whether the requirements are completely and appropriately implemented" (Spillner, Linz & Schaefer, 2014). The technical system design phase describes the overall architecture of the system and how each system and subsystem interface with each other. The corresponding testing tasks include the integration tests verifying that all components collaborate and communicate correctly with each other. All subsystems, behaviors, structures, and interfaces are defined during the component specification task, resulting in the testing criteria for the individual components themselves. Generally, programming begins after the above-mentioned development tasks have been completed and produce the individual components necessary to start testing.

**Component Level Tests**

Component testing is individual tests completed on specific objects, classes, modules, etc., independently of the system they will be integrated with. The components must be available, compiled, and executable in the test environment with both entry and exit level criteria available for both the functional and non-functional requirements.

**Integration Level Tests**

Integration testing focuses on the interfaces between the components that have passed the component test successfully and includes the operating system, file systems, database systems, hardware, and software both internally within the system as well as any external interfaces. Ideally, these tests would be completed incrementally as components and systems become available using stubs to fill in the gaps where needed, as testing all interfaces at once, such as in the "big bang" integration, tends to complicate the isolation of any defects resulting in increased time and costs needed for repairs.

**System Level Tests**

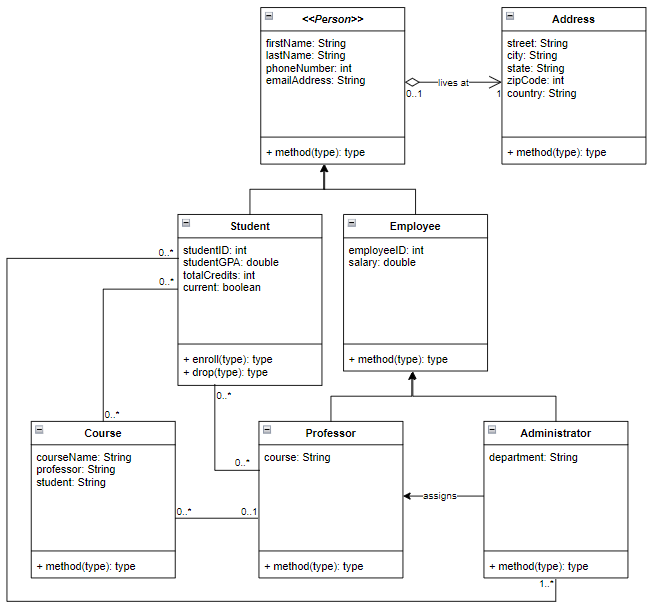
System testing focuses on how the overall system functions in accordance with requirements, specifications, and any other necessary standards. This is the final step before the system is passed to the user for acceptance testing and requires extensive documentation and verification before completion.

**Acceptance Level Tests**

Acceptance testing passes the completed software or system, its documentation, all necessary configuration items, forms, reports, and statistics to the user for final inspection and, hopefully, acceptance. This stage of testing must only be after all components, systems, and interfaces have been thoroughly tested, and all defects and failures have been corrected with confirmation and regression tests passed. If too many errors exist, the confidence in the software or system will be low, and the customer will be skeptical about any future deliveries.

**Figure 1.**

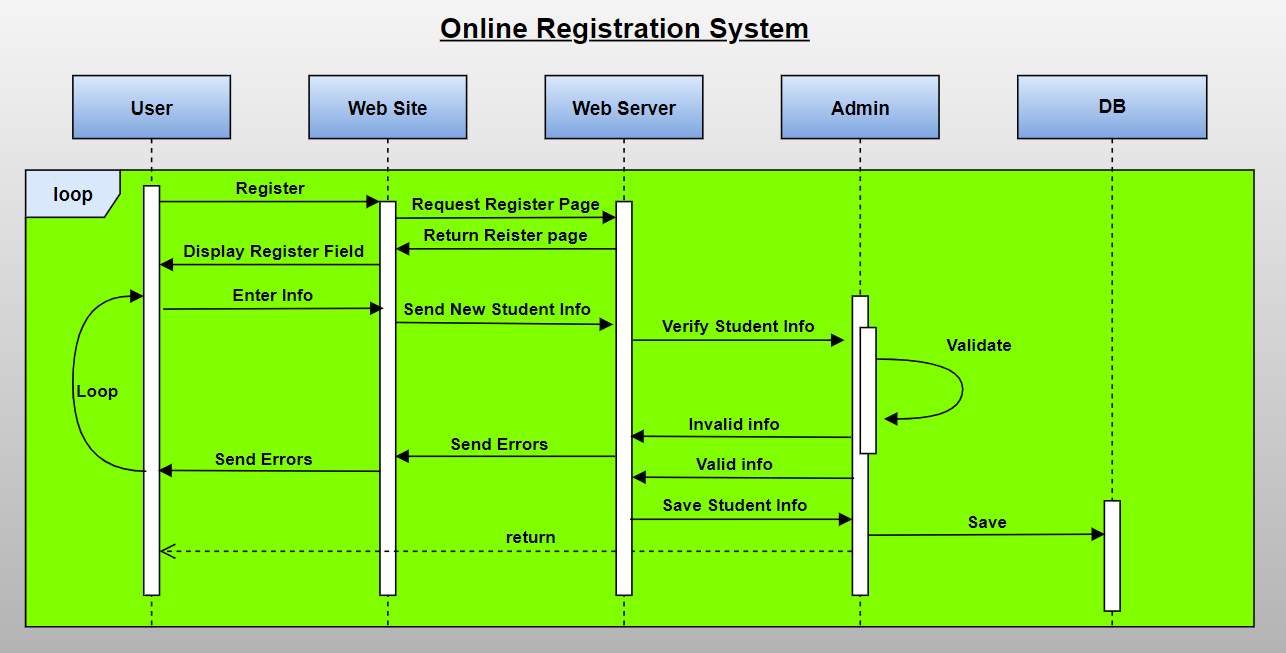
UML Class Model



*Note: The UML model shown above depicts the relationships between the different classes in the design.*

**Figure 2.**

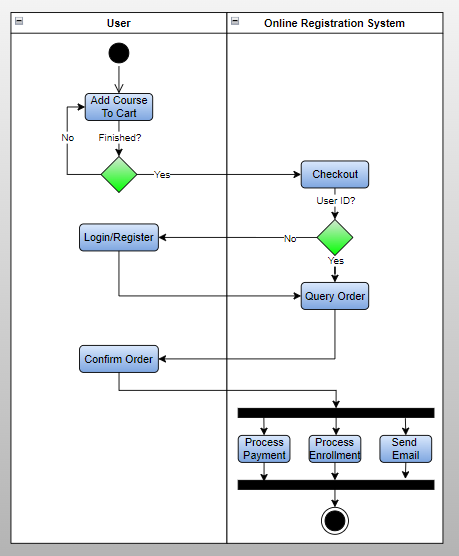
UML Sequence Diagram



*Note: The figure above depicts the sequence of events using UML for the online registration system.*

**Figure 3.**

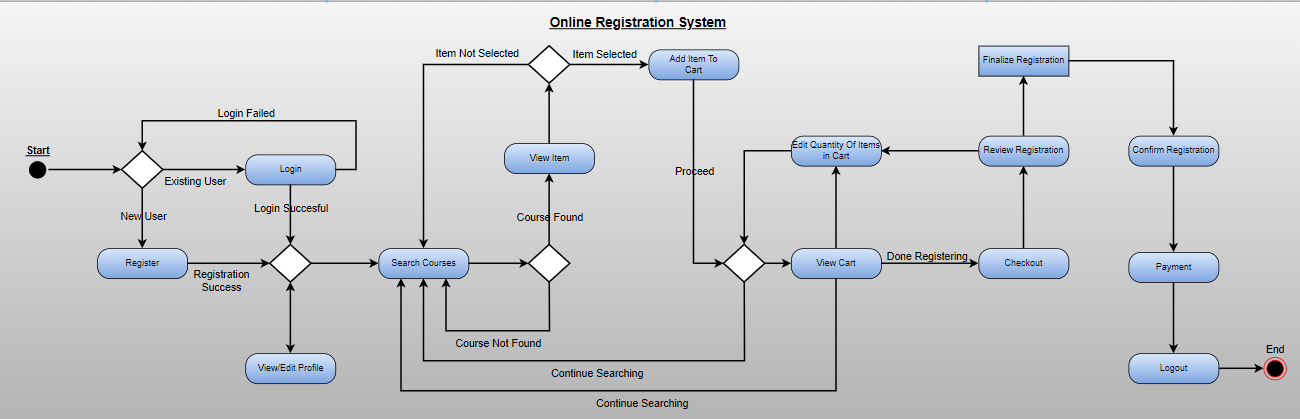
UML Activity Diagram



*Note: The figure above depicts the UML activity diagram associated with the online registration system.*

**Figure 4.**

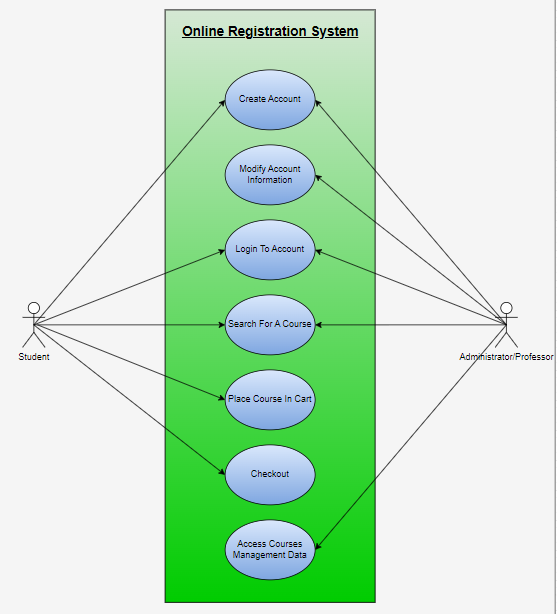
UML State Diagram



*Note: The figure above depicts the UML state diagram of the online registration system.*

**Figure 5.**

UML Use-Case Diagram



*Note: The figure above depicts a general use-case scenario diagram for an online registration system.*

**Conclusion**

This paper has discussed the importance of test planning and implementation for the overall success of the product and given the necessary UML models along with test level descriptions for implementation purposes. Testing is imperative for producing a quality product that will meet all requirements and specifications while avoiding unnecessary costs for rework later. Following the suggestions provided within this report will not only help to achieve system deployment but will ensure the highest possible return on investment.

**References**

Homès, B. (2012).*Fundamentals of software testing*. Wiley. Retrieved from <https://ebookcentral.proquest.com/lib/ashford-ebooks/reader.action?docID=1120766&ppg=123>

Spillner, A., Linz, T., & Schaefer, H. (2014). [*Software testing foundations: A study guide for the certified tester exam* (4th ed.)](https://ashford.instructure.com/courses/97026/modules/items/4915931). Rocky Nook.