**UML Design Modeling for Online Course Enrollment System**

Augustine Ugwu

The University of Arizona Global Campus

CST499: Capstone for Computer Software Technology

Dr. Charmelia Butler

November 19 2024

The purpose of this assignment is to translate the requirements from the Software Requirements Specification (SRS) document for the Online Course Enrollment System into a design. Using Unified Modeling Language (UML), I created several diagrams to capture both the static and dynamic aspects of the system. These models provide a comprehensive view of the system architecture and interactions, ensuring clarity for developers and stakeholders. Additionally, I discuss different levels of testing required to ensure the system meets its functional and non-functional requirements.

**UML Models**

**1. Class Diagram**

The class diagram represents the static structure of the system, showcasing its main entities, their attributes, and relationships.

A diagram of a computer program

Description automatically generated

**2. Use Case Diagram**

The use case diagram depicts the interactions between users (actors) and the system.

**Figure 2: Use Case Diagram**

* **Actors:** Student, Administrator, Instructor

A diagram of a tutorial

Description automatically generated

**3. Sequence Diagram**

The sequence diagram models the dynamic flow of operations for a specific scenario: a student enrolling in a course.

A diagram of a course

Description automatically generated

**4. Activity Diagram**

The activity diagram captures the workflow for course enrollment.

**Figure 4: Activity Diagram**

A diagram of a course

Description automatically generated

**5. State Diagram**

The state diagram models the state transitions for a course.

A diagram of a student's process

Description automatically generated

**Levels of Testing**

**1. Component Testing**

Component testing focuses on individual units of the system, such as classes or functions. For example, I would test methods like registerUser() and enrollCourse() in isolation to ensure they function as expected. This level identifies and resolves logic errors within a single module.

**2. Integration Testing**

Integration testing examines the interactions between components, such as how Student interacts with the Course entity during enrollment. This ensures that modules communicate effectively, and data flows correctly between them, as depicted in the sequence diagram.

**3. System Testing**

System testing validates the entire system against the requirements outlined in the SRS document. This includes testing the functionality of all use cases, such as course browsing, enrollment, and waitlist management, under realistic conditions.

**4. Acceptance Testing**

Acceptance testing ensures that the system meets the end-user requirements. This involves scenarios where students, administrators, and instructors interact with the system to confirm usability and performance. For instance, students must successfully enroll in courses or be added to waitlists.

By translating the requirements into UML diagrams, I created a detailed representation of the Online Course Enrollment System. These diagrams offer a clear view of the system's structure and behavior, making it easier to implement and test. The various testing levels—component, integration, system, and acceptance—ensure that the system functions as expected and meets user needs.