UML Design Modeling

CST499

Professor Rangitsch

Brian Ivers

2/10/25

A vital tool for planning and illustrating the composition and functionality of a software system is the Unified Modeling Language (UML). In this paper, I describe the class, use case, sequence, and activity diagrams that are available for modeling an online course registration system using UML. I also go over the several testing levels—Component Testing, Integration Testing, System Testing, and Acceptance Testing—that are necessary for confirming the system's dependability and functionality.

**UML Design Modeling**

**Class Diagram**

**A computer screen shot of a computer

AI-generated content may be incorrect.**

By outlining the important entities, characteristics, and connections, the class diagram illustrates the system's structural layout. The following are the main courses offered by the Online Course Registration System:

* **User (Super Class)**
  + Attributes: userID, password, name, phone, email
* **Student (Inherits User)**
  + Attributes: enrolledCourses, waitingList
* **Admin (Inherits User)**
  + Method: manageCourses()
* **Course**
  + Attributes: courseID, courseName, semester, maxEnrollment, enrolledStudents, waitlist
* **Enrollment**
  + Attributes: student, course, status
* **AuthenticationSystem**
  + Methods: registerUser(), loginUser()
* **CourseManagement**
  + Methods: listCourses(), enrollStudent(), dropCourse()

**Use Case Diagram**

**A diagram of a course

AI-generated content may be incorrect.**

The Use Case Diagram shows how system features and actors (students and administrators) interact:

* Student actions include registering, logging in, seeing courses, enrolling, dropping courses, and joining a waitlist.
* Admin actions include viewing course lists, managing enrollments, adding or removing courses, and logging in.

**Sequence Diagram**

**A diagram of a student

AI-generated content may be incorrect.**

The student-system interaction is depicted in the course enrollment sequence diagram:

1. The student signs in.

2. The student asks for the courses that are offered.

3. Courses are displayed by the system.

4. The student chooses a course and submits an enrollment request.

5. After verifying availability, the system adds the student to the queue or enrolls them.

6. The waitlist or enrollment status is verified by the system.

**Activity Diagram**

**A screenshot of a computer

AI-generated content may be incorrect.**

The steps in the Activity Diagram for dropping a course are as follows:

1. The student accesses their enrolled courses after logging in.

2. The student decides which course to discontinue.

3. The student is dropped from the course by the system.

4. The system looks for kids on the waitlist.

5. Students on a waitlist are notified to enroll if there is one.

6. When the student on the waitlist confirms their registration, the list is updated by the system.

The goal of component testing is to confirm that each system module or component operates as intended. Unit tests are usually used to test each component separately in order to find and address flaws early on. Prior to integration with other modules, this stage of testing makes sure that every unit operates as planned.

Integration testing assesses how various system modules or components interact with one another. This stage of testing makes ensuring that data moves smoothly across the system and that integrated units interact accurately. A variety of techniques, including hybrid integration testing, top-down, and bottom-up, can be applied to methodically confirm component interactions and dependencies.

To confirm that a system satisfies requirements, system testing evaluates the system as a whole. This kind of testing, which includes both functional and non-functional tests including performance and security testing, is carried out in a setting that closely mimics the production setup. The objective is to find any flaws that might appear when every part functions as a single unit.

The last stage of testing, known as acceptance testing, is carried out to ascertain whether the system satisfies user and business needs. End users or other stakeholders frequently participate in this testing phase to assess the usability and functionality of the system. It guarantees that the program satisfies requirements and is prepared for use. In addition to User Acceptance Testing (UAT), acceptance testing may also include contractual or regulatory compliance testing prior to the system's launch.

While the many testing stages guarantee that the system is secure, dependable, and useful, the UML diagrams offer an organized method for creating the online course registration system. System testing evaluates overall performance, acceptability testing verifies user happiness, integration testing guarantees module interactions, and component testing checks specific functions. We can create a reliable and effective course registration system by adhering to these guidelines.

*References:*

* Sommerville, I. (2019). *Software engineering* (10th ed.). Pearson.
* Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (9th ed.). McGraw-Hill Education.
* Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *The unified modeling language user guide* (2nd ed.). Addison-Wesley.
* Tsui, F., Karam, O., & Bernal, B. (2018). [*Essentials of software engineering*](https://uagc.instructure.com/courses/142034/modules/items/7241502)(4th ed.). Jones & Bartlett Learning.