**Revised UML Design Modeling Paper**

Jeremy Roosa

The University of Arizona Global Campus

CST 499: Capstone for Computer Software Technology

Instructor: Amr Elchouemi

January 18, 2022

**Introduction**

The following is an exciting depiction of the different UML models that capture the critical design and requirement aspects of the EasyEnroll Online Course Software platform. These models include class diagrams, sequence diagrams, activity diagrams, state diagrams, and use case diagrams. Although not all-inclusive, these diagrams will help tremendously in developing the software and understanding its architecture.

Figure 1.1 Revised UML Design Models for the EasyEnroll Online Course Software pt. 1

Diagram

Description automatically generated

Figure 1.2 Revised UML Design Models for the EasyEnroll Online Course Software pt. 2

Diagram

Description automatically generated

Figure 1.3 Revised UML Design Models for the EasyEnroll Online Course Software pt. 3

Diagram

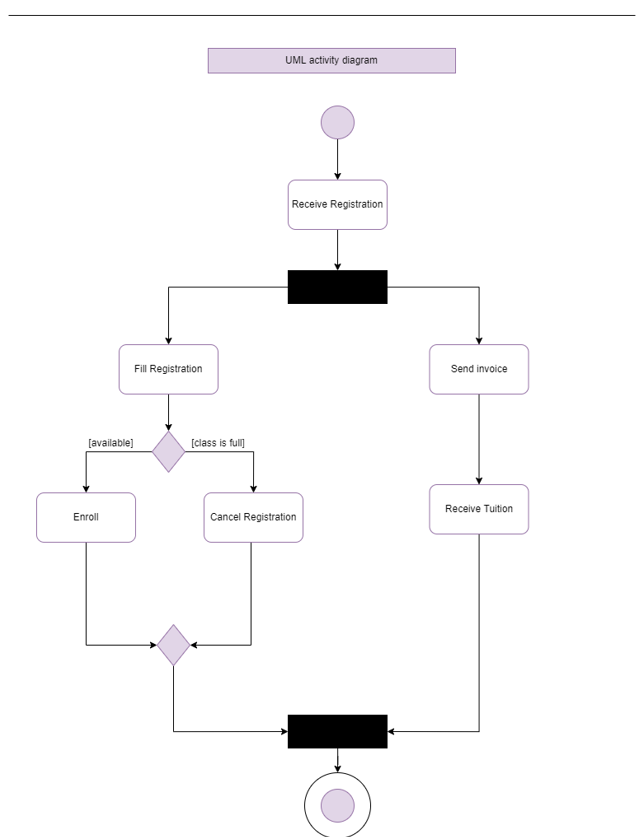
Description automatically generated

Figure 1.4 Revised UML Design Models for the EasyEnroll Online Course Software pt. 4

Diagram

Description automatically generated

Figure 1.5 Revised UML Design Models for the EasyEnroll Online Course Software pt. 5



Along with these UML models, it is essential for software developers to understand the intricacies associated with different levels of testing throughout software development. Unit testing and component testing ensure that individual parts of a system operate when run alone. Integration testing and system testing help test a system once all components are integrated, and the system can be run as a whole. The final testing stage includes acceptance testing, where users and authorization entities test the system to see if it meets all the agreed-upon requirements.

Overall, UML models, component testing, integration testing, system testing, and acceptance testing play a vital role in the design modeling and software system development processes.

**Discuss the details for Component testing**

Component testing is very critical to ensure all aspects of a system are functioning correctly and units can be eventually integrated. Each individual component is tested separately throughout component testing without integrating other units or components. Understanding the differences between component testing and unit testing is essential to fully understand this first step in the testing process. According to Hamilton (2021), in the article “What is Component Testing? Techniques, Example Test Cases,” “component testing is performed by testers. Unit testing is performed by the developers where they do the testing of the individual functionality or procedure. After unit testing is performed, the next testing is component testing” (para. 4). There must be a minimum number of components to be included, and exit criteria should be established for each component. Overall, component testing is crucial to the software development for a system and enables effective integration testing in later stages of development.

**Discuss the details for Integration testing**

After unit testing and component testing are complete, integration testing techniques can be implemented for the software being developed. Multiple components are brought together and tested in a group in integration testing. These various components are tested for their ability to work together to perform functionalities and for interoperability. According to Tsui, Karam, & Bernal (2018), in the course textbook “Essentials of software engineering (4th ed.),” “when dealing with very large software systems, functions may be integrated into a component. Many components are then brought together to form a system” (p. 213). There are various different types of integration testing that include big-bang, mixed, top-down, and bottom-up. Various testing patterns can be implemented, including backbone integration, layer integration, and client-server integration. Overall, integration testing is vital to the software development cycle and must be completed before system testing can be accomplished for the overall system to be evaluated.

**Discuss the details for System testing**

While integration testing is important to understand, system testing is one of the last steps in the testing process that software developers should consider. Quality assurance (QA) teams evaluate how an application’s components interact in the fully integrated application or system in system-level testing. The application is assessed to see if it meets all of its technical, business, functional, and non-functional requirements. According to Black (2017), in the article “System Testing,” “additionally, each individual type of system test reports relevant metrics of a piece of software, including Performance testing: speed, average, stability and peak Response times; Load testing: throughput, number of users, latency; and Usability testing: user error rates, task success rate, time to complete a task, user satisfaction” (para. 6). Many tools can be used throughout system testing to manage and automate test cases throughout the process. These automated tools can perform various functions on the system and measure performance and usability. Overall, system testing is vital throughout the software development and is necessary before acceptance testing can begin.

**Discuss the details for Acceptance testing**

After system testing is complete, acceptance testing can begin, and the software system can be reviewed entirely for the requirements and validation of the system. Acceptance testing is usually referred to as end-user testing in which the system or component is evaluated by the user, customer, or authorization unit. Acceptance testing can also be referred to as user acceptance testing, business acceptance testing, alpha testing, and beta testing. According to Craft (2021), in the article “Acceptance Testing,” “the main purpose of this test is to evaluate the system’s compliance with the business requirements and verify if it is has met the required criteria for delivery to end users” (para. 2). These testing strategies aim to encourage collaboration between the users and developers to discover if requirements have been met and formally agree the contract has been fulfilled. User stories can be utilized throughout acceptance testing as one of the final stages of the software development life cycle. Overall, acceptance testing is a vital part of the software development lifecycle and should be fully implemented to ensure software meets the desired requirements.

**Conclusion**

The various testing stages, including unit/component testing, integration testing, system-level testing, and acceptance testing, all play an important role in ensuring software is built according to functional and non-functional requirements. Firstly unit tests are run, followed by component tests in which individual components are tested separately. Secondly, integration testing procedures take these components and run them in parallel to test for interoperability and system integration capability. Thirdly, system-level tests evaluate the overall system and measure it for various performance criteria. Lastly, the software system is given to the user for acceptance testing and to ensure it meets the contract requirements. Overall, these different testing phases contribute to the software development life cycle and should be fully implemented throughout software development.

**References**

Black, R. (2017). System Testing. SearchSoftwareQuality; TechTarget. https://searchsoftwarequality.techtarget.com/definition/system-testing

Craft, J., (2021). Acceptance Testing. Tutorialspoint.com. https://www.tutorialspoint.com/software\_testing\_dictionary/acceptance\_testing.htm

Hamilton, T., (2021). What is Component Testing? Techniques, Example Test Cases. Guru99. https://www.guru99.com/component-testing.html‌

Tsui, F., Karam, O., & Bernal, B. (2018). Essentials of software engineering (4th ed.). Jones & Bartlett Learning.

‌

‌

‌