Week 2 - UML Design Modeling

Alex Nieves  
CST499: Capstone for Computer Software Technology  
Instructor: Amr Elchouemi  
10/10/2022

Software must be tested continuously throughout the development process to guarantee a high-quality end result. Effective testing may cut costs, speed up development, enhance maintainability, boost code quality, find and fix errors early in the development cycle, raise resilience, and guarantee that the client is getting a high-quality product that satisfies their requirements. Component testing, integration testing, system testing, and acceptance testing are the four tiers of testing (Spillner, Linz, & Schaefer, 2014). Software is broken down into functional pieces for component testing, which checks whether or not these components adhere to the necessary requirements. Integration testing is the process of testing groups of related components together to find issues with their interfaces and how they interact (GeeksForGeeks, n.d.). System testing "in particular concentrates on testing the functional and non-functional aspects of the software in a more thorough manner, including security, usability, performance, and compatibility" (Suffian et.al., 2016). Acceptance testing makes ensuring the software is functional in a production-level setting and satisfies the needs of the client.

Component testing

Component testing will take place for each of the many functionalities shown in the use case diagram in Figure 1 with reference to the Course Enrollment System. In order to display a student's schedule, dynamically generate the dropdowns that allow a student to search for courses, generate search results based on form input, add a course to a student's schedule, remove a course from a student's schedule, and add a student to a waitlist for a course, for instance, the software will have components that carry out each of the various functions. It is imperative to thoroughly test each class and function separately. Additionally, component-level testing will make sure that the code is error-free, manageable, efficient, robust, and performant.

Integration testing

Integration testing will also be required for the Course Enrollment System. Testers and developers can "identify defects and errors in the interaction between integrated components" by using integration testing (Ali et.al., 2018). Test cases will be created to assess how well the aforementioned components work together once they have all been properly tested on their own. Figure 4, for instance, shows the courses that must be taken in order for a student to enroll for a course. While individual components may pass component tests, the real test is whether or not control and data flow are maintained throughout all interactions.

System testing

The Course Enrollment System will need to pass a number of system tests when integration tests are finished, which determine if the website satisfies the functional and non-functional requirements. Does the website, for instance, have every feature listed in the SRS? Is the Course Enrollment System website accessible, secure, quick, and easy to maintain? At this stage, the testers and developers must consider the system from the viewpoint of the user and test it in a setting that is as close to the production-level as possible.

Acceptance testing

The Course Enrollment System will undergo acceptance testing at the end. Students will be involved in this process to make sure that it offers them all the functionality they require to swiftly browse, register for, and drop classes. Administrators will also need to comment on whether or not they can alter course offers and generate reports fast and simply. To guarantee that every user will have a great user experience, acceptance testing will be conducted across a variety of operating systems and devices. Component, integration, system, and acceptance testing should be carried out to help the development team deliver a high-quality product as quickly and effectively as feasible.

Figure 1 Use Case Diagram

Diagram

Description automatically generated

Figure 2 Activity Diagram

Diagram

Description automatically generated

Figure 3 Sequence Diagram

A picture containing chart

Description automatically generated

Refrences:

Ali, S., Imran, M., Hafeez, Y., Abbasi, T. R., Haider, W., & Salam, A. (2018). Improving Component Based Software Integration Testing Using Data Mining Technique. 2018 12th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS), Mathematics, Actuarial Science, Computer Science and Statistics (MACS), 2018 12th International Conference On, 1–6. https://doi-org.proxy- library.ashford.edu/10.1109/MACS.2018.8628368

GeeksForGeeks.com. (n.d.). Software Engineering: Control Flow Graph (CFG). Retrieved April 18, 2020 from https://www.geeksforgeeks.org/software-engineering-control-flow-graph-cfg/

Spillner, A., Linz, T., & Schaefer, H. (2014). Software testing foundations: A study guide for the certified tester exam (4th ed.). Rocky Nook.

Suffian, M., Rizal, F., Loo, Aman, Bajuri, Misek, Fistein, Zavoral, Koller, Gala, Perez, G., Ruland, & Fohler. (n.d.). *[IEEE 2016 IEEE conference on open systems (ICOS) - Langkawi Island, Kedah, Malaysia (2016.10.10-2016.10.12)] 2016 IEEE conference on open systems (ICOS) - software capability rating using system testing scores*. https://ur-booksc-me.translate.goog/book/64540594/0e2008?\_x\_tr\_sl=ur&\_x\_tr\_tl=en&\_x\_tr\_hl=en&\_x\_tr\_pto=sc