**Week 2 – UML Design Modeling**

Adam E. Svatek

The University of Arizona Global Campus

CST499: Capstone for Computer Software Technology (CSF2415A)

Professor Joseph Rangitsch

May 13, 2024

**Unified Modeling Language (UML) Diagrams**

UML diagrams are excellent tools used to visually convey information, concepts, and relationships of users, objects, classes, and other information in software development. For example, the software requirement specifications for a software system used to register student for college classes would use a class diagram, sequence diagram, activity diagram, and user case diagram.

Class diagrams (Figure 1) are used in object-oriented programming to show different classes of objects, their attributes, and how they relate to other classes of objects. In this instance, the student can have zero to many schedules which contain zero to many courses. The attributes are shown below each class of object. Courses can belong to zero to many different schedules. Each schedule will belong to one and only one student.

Diagram

Description automatically generated

**Figure 1: Class Diagram**

Sequence diagrams (Figure 2) are used to show a sequence of events that take place and the actors involved. In this case, the user, or student, will create an account, log into this account, sign up for a class, and then log out. The series of events occur from top to bottom, and left to right.

Diagram

Description automatically generated

**Figure 2: Sequence Diagram**

Activity diagrams (Figure 3) show a more simplified flow path of events that are to take place. With this diagram, the user starts performing a search for classes. They either find the class that they are looking for or they don’t. The diagram goes different directions from here. The user could end up viewing the details for the class that they were looking for and either register for it or continue to search. Ultimately, the user will end their session by signing out of the system.

**Diagram

Description automatically generatedFigure 3: Activity Diagram**

Use case diagrams (Figure 4) are simplistic diagrams that show how the user(s) will interact with the system. The users are outside of the software system, so they are outside od the box depicted. The actions in ovals inside the system are attached to the users that will interact with these features. The student will be able to perform all these actions. The college administrator, however, is limited to what actions they can perform. In this example the college administrator can perform all of these actions, except to log in as the student.

Diagram

Description automatically generated

**Figure 4: Use Case Diagram**

**Methods of Software Testing**

Using a traditional software design methodology, such as the waterfall method, developers will develop software in the order shown on the left side of the general V-model below. Along with each of these development steps comes a specified level of testing. The development of this software approach ends with the programmers writing the code. The testing begins with the code and works its way back up the process ending with testing the software to ensure it meets the requirements definition.

Diagram

Description automatically generated

**Figure 5: General V-Diagram**

Component testing has many other names such as unit testing, module testing, function testing, procedure testing, and object testing. The term “component testing” can be used more generally as testing that is performed by the developers of the software or by their fellow developers. When performing component testing, the tester will test the smallest testable piece of code that can be tested. Ideally these components will be tested in isolation from other systems and components to ensure the testing is specific to that component and that the errors that are found are within that component and not an outside source. In order to isolate the component being tested, developers will mock the interfacing software or components that would normally interact with that component.

Integration testing is the next level of testing moving up the right side of the V-model. This level of testing is associated with the technical system design or sometimes known as the architecture design phase. In integration testing, the pieces of software that were tested during component are tested with other pieces of software to ensure that they function and communicate with each other correctly. These tests will show how groups of components or broader subsystems of the software work with each other. Outcomes of these tests will influence changes in the technical systems of the design system which may then be made throughout additional phases of the development process.

Following integration testing comes system testing. This is where the functionality of the entire software system is tested. It is testing to see how well it communicates with other software systems, hardware, browsers, and other external systems. The findings of these tests will be used for improving the functional system design of the software. Additional changes may need to be may in other levels of the software design to support these changes.

Acceptance testing is the final rung of the software testing ladder. Both functional and non-functional requirements are tested here by stakeholders and other testers who view the software as a user rather than a developer. The software is ideally tested in the type of environment where it is expected to run to get the best results. In addition to making sure the software runs in its targeted environment, testers will verify that the requirements of the software are met. Changes may be made at all levels to accommodate the findings of acceptance testing. Once the software passes this level of software testing it can be delivered to the customer.

**References**

Adam, J. (2021, September 24). What is the V-model software development methodology? K&C. https://kruschecompany.com/v-model-software-development-methodology/

GeeksforGeeks. (2024, March 12). Class Diagram Unified Modeling Language (UML). GeeksforGeeks. https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/

Sommerville, I. (2016). Software Engineering (10th ed.). Pearson.

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