**UML Design Modeling**

Jaime Aviles

University of Arizona Global Campus

CST 499

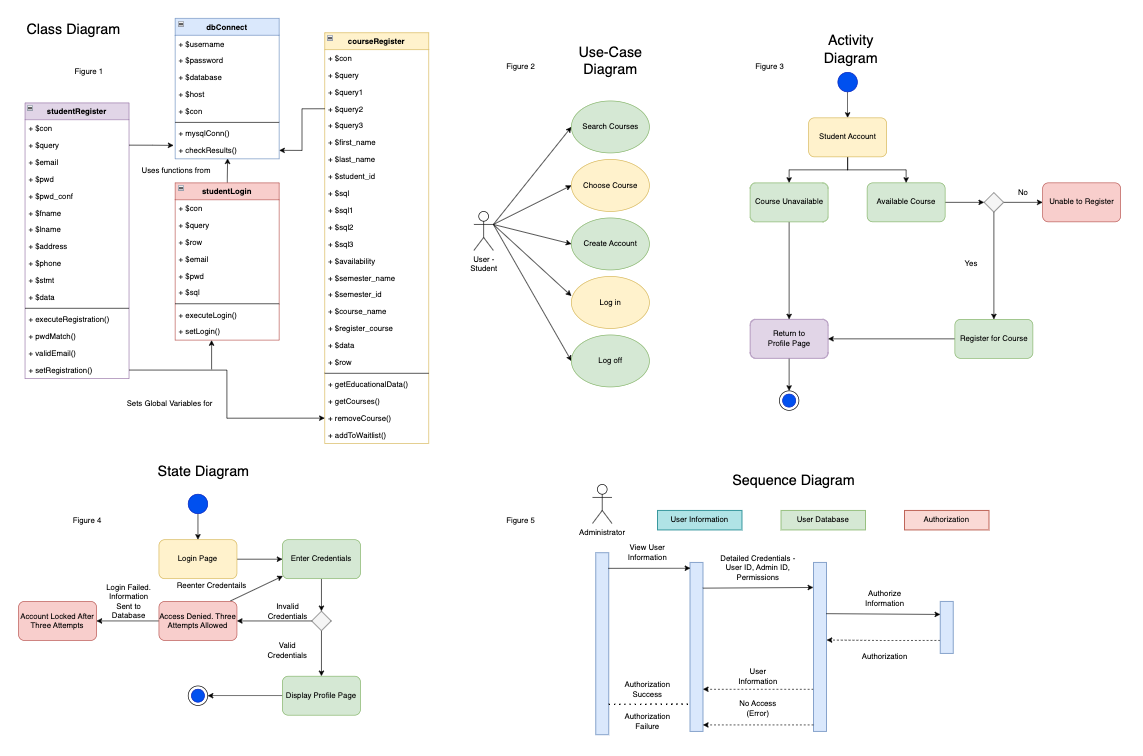
Dr. Amr Elchouemi

12/20/22

**UML Modeling**

**Introduction**

System design is crucial to building a successful system that does exactly what the user needs. The details of the system design are just as crucial to make sure all the system requirements are functioning to the stakeholder’s specifications. That is why Unified Modeling Language (UML) designs along with testing methods are key in the development and deployment of good usable software. In this paper, we will discuss the importance of UML designs and software testing to get a better understanding of their importance throughout the software’s development.

****

**Figure 1**

Class diagram. Depicts the relationship between the class and its subclasses (Sommerville, 2016).

**Figure 2**

Use-Case Diagram. Depicts an actor’s interaction with and aspect of the system (Tsui, Karam, & Bernal, 2018).

**Figure 3**

Activity Diagram. Depicts the activity of certain stimuli in the system (Sommerville, 2016).

**Figure 4**

State Diagram. Depicts the state of the system after a stimulus is triggered (Sommerville, 2016).

**Figure 5**

Sequence Diagram. Depicts the flow of messages from one object to another. (Tsui, Karam, & Bernal, 2018).

Component Testing is the first level of testing where the component specifications are tested after the programming of the software. These tests are based on the component requirements and design and are centered on the software’s class model. Figure 1 depicts the UML class model, where the primary class is built from its subclasses and is centered around the website’s user structure. Variables and modules (functions) are a part of each class certain classes are reused to fulfill the tasks of a separate class (Sommerville, 2016). The major factor of component testing and the software classes is that classes are tested individually from each other to isolate any impact from outside influences, also known as loose coupling (Spiller, Lintz, & Schaeffer, 2014).

Integration Testing amalgamates the structural units and subsystems of the software and processes them to make sure that they work properly with each other (Spiller, Lintz, & Schaeffer, 2014). Figure 5’s sequence diagram depicts how the integration of the subsystems works. The administrator will interface with the system and the system interfaces with the database (Sommerville, 2016). Integration Testing will further expose defects in the software but will be

more concerned with the individual components of the program. Incompatibility between modules, bad data flow, inadequate exception handling, poor hardware compatibility, and program logic differences errors are checked in this process (*Integration Testing - Javatpoint*, n.d.)

System Testing is focused on the software functions from the user’s standpoint. The Use-Case diagram in Figure 2 depicts how the user would interact with the course registration page for the website system. System Testing ensures that all the functional requirements are actually working as designed and makes sure they meet the stakeholder’s standards. Documentation, such as guides and risk analysis during the system testing is also commonly included (*Learn V-Model in Software Testing*, n.d.).

Lastly, in acceptance testing, all tests are performed by specific testers that involve the customer, who will be responsible for the entire testing in certain situations. Acceptance tests are typically manually done and involve almost no automation to find defects (*Learn V-Model in Software Testing*, n.d.). There are four common testing types in acceptance testing, which are contract acceptance testing, user acceptance testing, operational acceptance testing, and field testing. Contract acceptance testing makes sure that the software functions to the specification of the contract and must follow a method that is as plainly defined as possible. User acceptance testing follows a process that involves various user groups. Since different user groups may have different viewpoints on the system’s expected outcomes, user acceptance testing is imperative to determine what can be changed in the next development cycle. Operational acceptance testing involves backup/restore, disaster recovery, security vulnerabilities, and user management and must be accepted by the stakeholder’s system administrators. Lastly, field testing is external

environment testing that focuses on identifying the unknown influences from the user’s environment, that the development team cannot see (Spiller, Lintz, & Schaeffer, 2014).

**Conclusion**

A system is only as good as its developers design it to be. With UML models that describe various aspects of the system's behavior and the testing of its crucial functions, designers can create a system that achieves the stakeholder’s goals and allow it to be effective in its usability to its user base. Without these two aspects of system design, the software would not be worth the time and effort taken to get it to production, thus failing the project.

**References**

*Learn V-Model in Software Testing*. (n.d.). Www.tutorialspoint.com. Retrieved October 8, 2022,

from https://www.tutorialspoint.com/learn-v-model-in-software-testing

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

Tsui, F., Karam, O., & Bernal, B. (2018). [Essentials of software engineering](https://ashford.instructure.com/courses/111216/modules/items/5653165) (4th ed.). Jones & Bartlett Learning.

*What is Integration Testing?* (2021, October 28). Codecademy News. *Integration Testing –*

*javatpoint*. (n.d.). www.javatpoint.com. https://www.javatpoint.com/integration-testing