**The UML Design Modeling Paper**

Kevin Cox

CST 499: Capstone for Computer Software Technology

Instructor: Amr Elchouemi

March 13, 2022

**Introduction**

The V-Model is a simple diagram that maps out the process behind Software Development Life Cycle (SDLC). Beginning from the top left and moving down, the V-Model consists of defining requirements, designing the functional system, designing the technical system, specifying the components, and at the bottom of the V-Model is programming. Once all the requirements and design elements are decided and programmed, the V-Model moves up and to the right which consists of four tests which are, in order, component, integration, system, and acceptance testing.

**Component Testing**

Starting with component testing, these tests are to verify that the component elements such as requirements and designs that were defined on the left side of the V-Model have been met (Spillner, Linz, & Schaefer, 2014). The component tests examine the smallest chunks of code individually before moving on to the next set of tests along the V-Model, which is the integration tests. A white box testing style is typically used in this phase due to the testers having access or knowledge of the source code while testing. Static testing is very important in this phase as it focuses on looking for errors in the code without executing the code (Hamilton, 2022).

**Integration Testing**

The integration testing phase begins to combine the smaller components into their intended larger chunks of code or whole functions. Once the smaller component elements are known to function properly, debugging the combinations of components is the next hurdle along the V-Model. The integration tests align with the technical system design aspect on the left side of the V-Model and looks for the positive and negative collaboration between the components as well as reveal deficiencies in the interfaces and connections between the components. This type of testing typically uses four types of strategies which are top-down, bottom-up, ad hoc, and backbone integration tests (Spillner, Linz, & Schaefer, 2014).

**System Testing**

Following the component and integration tests, the next phase is the system tests. These tests are on the same level as the functional system design, and this is exactly what these tests do. They consist of testing that the system created matches what requirements the customer specified as well as that the users can perform the functions they need from the system. Once the component and integration tests are completed, the system is fully built and is now tested in environments as close to what the system will function in (Spillner, Linz, & Schaefer, 2014). This will allow the best-case scenario to determine if the system will function properly for the customer and users. This phase relying heavily on thorough definitions of what functions the system should be able to perform. Without these requirements comprehensively defined, all the good and bad aspects of the system would technically be defined as the systems functions which would not create a great system.

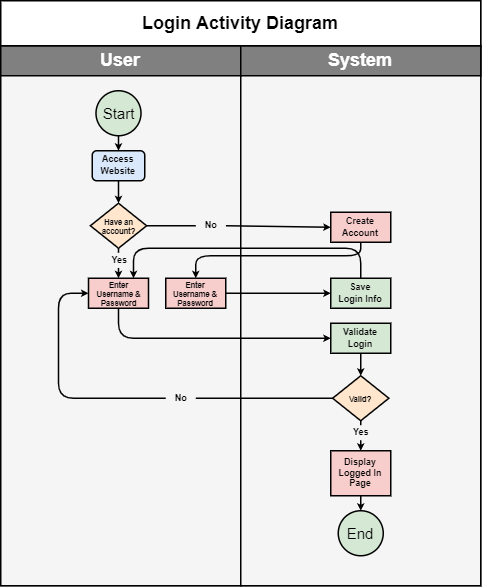
**Acceptance Testing**

Last up in the V-Model is acceptance testing which is the chance for the customers and users to “test drive” the software created to determine if it meets the needs that they requested. Contract acceptance testing is typically required for the customer to ensure testing by themselves or their employees (users) to see if the system functions can meet all the specifications that were originally established. This phase can also include field testing to determine how the software will perform in differing environments (Spillner, Linz, & Schaefer, 2014). These user acceptance tests can also be carried out by members of the companies IT department who will likely maintain the system once it is implemented.

**Conclusion**

The V-Model is a simple and effective method to walk through the Software Development Life Cycle of a system. Starting from the top left of the “V” and working along its length is the best practice to take a proposed software idea through the design, creation, and testing phases until it is a finished product, ready for deployment. The testing stages begin with the smallest piece of code and then combine them after each phase of testing is complete before beginning more tests. This process continues until the product is built and is then tested by the customers and users before getting their approval and the software being finished. This model is highly effective and lays out a direct path of where to move the software development from beginning to end.

**Figure 1 – Login Activity Diagram**



Note: This is a Unified Modeling Language (UML) which shows the interaction between the user actions and the systems actions. This model shows the Activity diagram of the Login system.

**Figure 2 – Course Enrollment Activity Diagram**

Diagram

Description automatically generated

Note: This model shows the Activity diagram of the Course Enrollment system.

**Figure 3 – Use Case Diagram**

Diagram

Description automatically generated

Note: This model shows the different tasks that different roles will be able to perform within the system.

**References**

Hamilton, T. (2022). *What is Static Testing? What is Testing Review?* Retrieved from <https://www.guru99.com/testing-review.html>

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