# UML Design Modeling

Benjamin Brown

The University of Arizona Global Campus

CST499: Capstone for Computer Software Technology

Professor Joseph Rangitsch

February 24, 2025

**Developing UML Models and Discussing Testing Levels**

***Introduction***

Based on the Software Requirements Specification (SRS) document developed in Week 1, this paper presents UML models capturing the key aspects of the system's design and requirements. There are several types of diagrams to choose from, such as Class Diagrams, Sequence Diagrams, Activity Diagrams, State Diagrams, and Use Case Diagrams. Several levels of software testing are supported by these models, including component testing, integration testing, system testing, and acceptance testing. A description of the role that each UML model plays during each phase of testing will be provided.

***Component Testing***

# Objective:

# In component testing, each module or component is tested individually to ensure that it performs as expected before being integrated.

# Relevant UML Models:

1. **Class Diagrams:**

# Diagrams such as these illustrate a module's internal structure, showing its attributes, methods, and relationships. Testers can identify dependencies and validate individual component designs with their help. According to Briand and Labiche (2002), classes can be divided into diagrams; these diagrams depict the structural composition of classes within a module, encapsulating attributes, methods, and relationships" (p. 2).

# Sequence Diagrams:

# In a Sequence Diagram, objects interact within a single component. Using these diagrams, you can analyze method calls and message flows to ensure that internal communication is working properly. According to Briand and Labiche (2002), sequence diagrams depict the sequential flow of interactions among modules (p.3).

# Activity Diagrams:

Diagrams of this type depict how a module's workflow works, describing a component's sequence of operations. By doing so, potential procedural issues can be identified. It is Briand and Labiche's (2002) observation that "Activity diagrams encapsulate the procedural logic and workflow within a module." (p. 4).

***Integration Testing***

# Objective:

# In integration testing, data flow and communication integrity are verified between different modules.

# Relevant UML Models:

1. **Sequence Diagrams:**

# A sequence diagram illustrates how modules interact. Diagrams like these illustrate how messages are exchanged between modules, making it easier to identify communication issues or integration problems. Mohanan (2022) explains that integration testing reduces bugs, errors, and issues in the interaction between integrated units, components, or modules.

# Class Diagrams:

# Integration Testing continues to benefit from class diagrams, since they depict the relationships between classes across modules. Identifying integration challenges is made easier through this process. As Mohanan (2022) points out, "Analyzing class dependencies can expose integration challenges" (paragraph 2).

# Use Case Diagrams:

Using these diagrams, integration test cases are formulated based on interactions between users (actors) and the system. Briand and Labiche (2002) assert that "Use case diagrams delineate the interaction between users and the system, guiding the formulation of test cases.".

***System Testing***

# Objective:

# The purpose of system testing is to ensure that the entire system meets the functional and non-functional requirements outlined in the SRS document.

# Relevant UML Models:

1. **Use Case Diagrams:**

To ensure comprehensive coverage of a system, Use Case Diagrams are essential for defining test scenarios. According to Briand and Labiche (2002), "Use case diagrams define test scenarios based on user interactions." (p. 6).

# State Diagrams:

# These diagrams illustrate how a system behaves under various conditions, validating state transitions. As part of the System Testing process, they are integral to modeling the behavior of the system. A state diagram provides a graphical representation of a system's transitions and states (Briand and Labiche, 2002, p. 7).

# Activity Diagrams:

Activity Diagrams remain relevant for System Testing, providing a visual representation of system functionality and assisting testers in comprehending the flow of testing activities. "Activity diagrams provide a visual representation of system functionality and help testers comprehend activity flow," Briand and Labiche (2002) write.

***Acceptance Testing***

# Objective:

# During acceptance testing, the software is validated against user and business requirements to ensure it meets stakeholder expectations.

# Relevant UML Models:

1. **Use Case Diagrams:**

In order to capture user requirements and define acceptance criteria, Use Case Diagrams are crucial. As a result, acceptance test cases can be crafted based on them. The use case diagram plays a pivotal role in capturing user requirements and defining acceptance criteria, according to Mohanan (2022).

# Activity Diagrams:

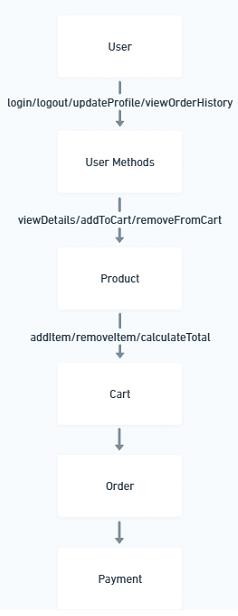
# Stakeholders can visualize system behavior using Activity Diagrams, which model user workflows and scenarios. By doing so, the usability and user experience of the software can be validated. Briand and Labiche (2002) emphasize that "Activity diagrams model user workflows and scenarios" (p. 9).

# Class Diagrams:

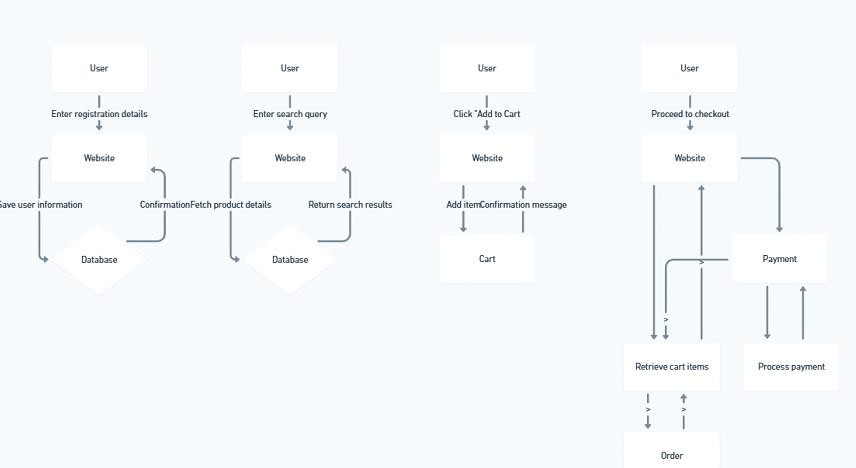
As part of Acceptance Testing, class diagrams may also be used to validate business rules or data structures. Briand and Labiche (2002) explain that class diagrams can be used to validate business logic and verify data structures.

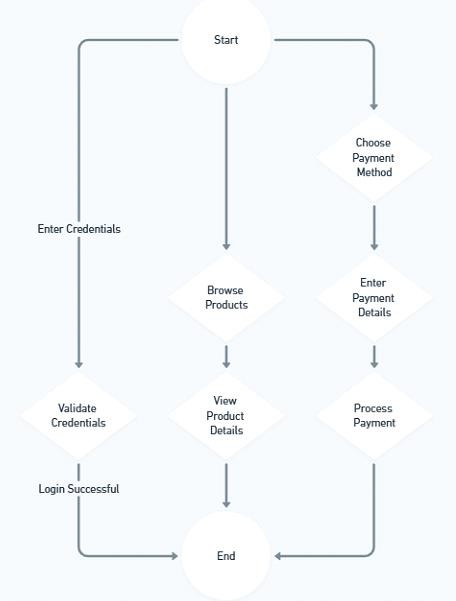
**Diagrams Examples Based on Week 1 Requirements**

⦁ **class diagrams**



⦁ **sequence diagrams**

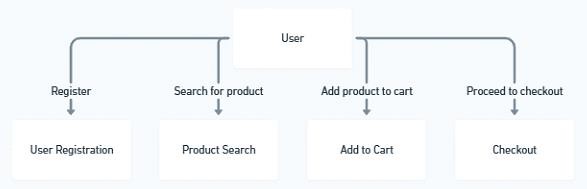
****⦁ **activity diagrams**



⦁ **state diagrams**



⦁ **use case diagrams**

****

**Conclusion**

Each UML model plays a critical role in expediting the testing process by offering visual representations of system design and behavior. These models ensure that testing is thorough, efficient, and aligned with the SRS document's requirements. By incorporating Class Diagrams, Sequence Diagrams, Activity Diagrams, State Diagrams, and Use Case Diagrams, teams can effectively identify and resolve issues at different testing levels, ensuring a robust and reliable software product.

**References**

Briand, L., & Labiche, Y. (2002). A UML based approach to system testing. *Software and Systems Modeling*, 1. <https://doi.org/10.1007/s10270-002-0004-8>

Mohanan, R. (2022, November 21). What Is Integration Testing? Types, Tools, and Best Practices. Retrieved from [https://www.spiceworks.com/tech/devops/articles/what-is-integration-](https://www.spiceworks.com/tech/devops/articles/what-is-integration-testing/)

[testing/](https://www.spiceworks.com/tech/devops/articles/what-is-integration-testing/)