Week 3: Interactive Assignment, Part III

System Modeling and Architectural Design

Eric Vara

The University of Arizona Global Campus

CST 301: Software Technology & Design

Professor John Howerton

January 19, 2023

**Diagram

Description automatically generatedSystem Modeling and Architectural Design**

**Diagram

Description automatically generated**

**Differences Between Applying Different System Models**

Essential viewpoints in software engineering and systems design include context, interaction, structure, and behavior modeling. To comprehend and create complex systems, various models are used since each one offers a unique perspective on the system under development. Interaction modeling focuses on how components interact with one another, structure modeling offers a thorough perspective of the components, and behavior modeling captures the system's changing behavior over time. Context modeling offers a high-level view of the system's surroundings.

Context modeling offers a broad perspective of the environment surrounding the system, including outside variables that affect the system's behavior. The focus of interaction modeling is on the relationships and interactions that exist between the various system components. Structure modeling offers a thorough perspective of the system's elements, including a look at their internal organization and interrelationships. Behavior modeling offers a dynamic perspective of the system, capturing how the parts work together and behave over time in response to alterations in the outside world or the actions of other parts.

**The Model View Controller**

Software engineers utilize the Model-View-Controller (MVC) design pattern to separate the presentation of information from how the user interacts with it. There are three primary parts to the pattern:

**Model**: Represents the application's core data and business logic. The Business Logic Layer or the Data Access Layer would normally be the counterpart of the Model component in a layered architecture.

**View**: Gives the viewer a visual representation of the data. The View component would normally match the Presentation Layer in a layered architecture. The View component would normally be built on the client side in a Client-Server architecture.

**Controller**: As a bridge between the Model and the View, the Controller manages user input and coordinates updates to both the Model and the View. Layered Architecture is a type of

By preventing changes to one component from affecting the other components, the MVC pattern's separation of responsibilities enhances the application's maintainability, scalability, and testability. Both the View and the Controller can be updated independently of the Model component, and the Model and Controller can be updated independently of the View.

In a web-based application, for instance, the Model component might stand in for a database of products and the information related to them, the View component might represent the user interface through which the user is shown the product information, and the Controller component might manage user input, such as adding or removing products from the database.

**5 Unified Modeling Languages**

Software system design is represented graphically using the Unified Modeling Language (UML). Different facets of software systems, such as their behavior, structure, and interconnections, are modeled using UML diagrams. Depending on the system's needs and the part of the system being modeled, a particular UML diagram will be utilized. The following are five popular UML diagrams, along with illustration uses for each:

An **activity diagram** is used to depict the movement of activities within a system. It is used to simulate the system's dynamic behavior, including action and decision sequences and control flow. The process flow for a bank's loan approval system, comprising the many phases involved in assessing a loan application and reaching a loan decision, can be modelled by a team using an activity diagram.

**Use Case Diagram**: A use case diagram is used to show how actors and the system interact. In addition to use cases and the connections between actors and the system, it is used to explain the functional requirements of the system. Example: To represent the interactions between the different players engaged in an airline's ticketing system, such as customers, agents, and the ticketing system, a team can use a use case diagram.

A **sequence diagram** is used to show how different objects or system components interact with one another. It is used to explain how the system behaves across time, including the message exchange patterns between objects and the chronological order in which messages are issued and received. A team can use a sequence diagram to represent the interactions among the various parts of a medical diagnostic system, such as those involving the patient, the medical equipment, and the doctor.

**Class Diagram**: In a class diagram, the relationships between the classes in a system are shown. It is employed to define the system's static structure, which includes the classes, their properties, and their connections to other classes. An e-commerce system's classes, such as the ones for customers, orders, and items, can be modeled using a class diagram by a team.

**State Diagram**: A state diagram is a visual representation of a system's behavior in response to an event. It is used to define the system's potential states as well as the changes that can occur between them, including the circumstances that set off the changes as well as the outcomes of the changes. As an illustration, a team can create a state diagram to represent the different states that a vending machine might be in, such as waiting for a customer, dispensing a product, and displaying an error message.

**Behavioral Modeling**

A method for describing how a system behaves in response to events is called behavioral modeling. While event-driven modeling concentrates on the occurrences that cause changes in the state of the system, data-driven modeling concentrates on the data that drives the behavior of the system. Both modeling approaches can be used to explain how a system behaves and, depending on the needs of the system, are suitable in certain situations. Data-driven modeling and event-driven modeling are the two basic categories of behavioral modeling (state diagram).

Data-driven modeling: In data-driven modeling, the values of the system's inputs and internal data influence how the system behaves. The model specifies a set of interactions between the system's inputs and outputs, and by resolving these relationships, the system's behavior is ascertained. The information that underlies the system's action is the focus.

Event-driven modeling (state diagram): In event-driven modeling, the occurrence of events controls the system's behavior. The model outlines the possible states for the system, as well as potential events and the resulting transitions between states. The emphasis is on the occurrences that cause changes in the system's state.

**Generic Models of Application Systems**

An application system known as a transaction processing system (TPS) processes transactions, the fundamental units of work that cause a change in the state of the system. TPSs are built to ensure the integrity of the data being processed while handling massive quantities of data and numerous transactions quickly.

Language Processing System (LPS): An application system that processes and modifies language data is known as an LPS. An LPS's primary objective is to transform one language into a different way of presenting that language, such as through transcribing spoken language into writing or translating a document from one language to another.

**Work Cited**

Koushik, B. (2022). What is the generic process model in software development? Educative. Retrieved January 19, 2023, from <https://www.educative.io/answers/what-is-the-generic-process-model-in-software-development>

Cherry, K. (2022, March 6). How is behavioral analysis used as a treatment? Verywell Mind. Retrieved January 19, 2023, from <https://www.verywellmind.com/what-is-behavior-analysis-2794865#:~:text=1%20Behaviorism%20is%20a%20branch,used%20to%20shape%20human%20behavior>.

Hernandez, R. D. (2021, April 20). The model view controller pattern – MVC architecture and Frameworks explained. freeCodeCamp.org. Retrieved January 19, 2023, from <https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/>

Osis, J., &amp; Donins, U. (2017). Unified modeling language. Unified Modeling Language - an overview | ScienceDirect Topics. Retrieved January 19, 2023, from <https://www.sciencedirect.com/topics/computer-science/unified-modeling-language>