Victor Ngetich

CS 255

4/4/25

**Model Application**

## 2. Process Model Application

To model the Driver Pass project, a data flow diagram (DFD) and activity diagram modeling strategy would use DFDs and activity diagrams as the modeling tools. The diagrams would be used to represent the sequence of data flow and steps of operation in the system environment.

For example,

The DFD presents the entire process of booking reservation from student requests to system confirmation activities and data flows between students, system and instructors.

The activity diagram of the Online Exam Process shows students initiating exams, which lead to question resolution before generating results, contributing to system records.

This process will provide steps to uncover flow problems and operation bottlenecks

## Object Model Application

UML class diagrams would be designed via object modeling in order to find out the system entities and their associated components.

Examples:

Student Class has the address and name along with payment facilities and contact information. Methods: register(), bookLesson(), takeExam().

Lesson Class consists of date, time, instructor, car, and package templates as attributes. Methods: schedule(), cancel(), modify().

The properties of the exam class are defined, relation by exam ID, questions, score, status. Methods: start (), submit(), grade().

A clean design by this method enables reuse of code within the system.

## Process and Object Model Comparison

**Advantages:**

• Process models such as BPMN or DFDs provide a clear graphical representation of the DriverPass process. It allows stakeholders ranging from developers and administrators to even the end-users to have an easy understanding of the step-by-step processes involved in license applications, renewals, and tests. This type of graphical simplicity allows a common understanding of the functionality of the system.

• Process models allow both conceptual and visual comprehension of workflow structures. This means that they are easy to understand, and they alert people to the order of operations.

• The process map of flow instantly identifies probable inefficiencies and bottlenecks. A process map, for example, can reveal too much waiting time in document verification or payment processing, and focused optimization efforts can be launched.

**Disadvantages:**

* May fail to represent data structures accurately: Process models are wonderful to use in representing data flow, but not to use to represent data relations, e.g., an object model will demonstrate.
* Complexity in Large Systems: Process models in themselves would grow too unwieldy in a system like DriverPass with multiple hundreds of related processes and streams of data. Complexity will render them hard to read and control, and be hard to trace changes and updates.
* Constrained Data Structure Representation: Process models focus on activity flow and data change with minimal concern for the underlying detailed data structures. Data stores are illustrated to represent data repositories, yet inter-data entity relationships are not so explicitly expressed as in object models.

**4.2 Object Modeling:**

**Advantages:**

• Allows to portray complicated data relationships. There are numerous such as "Applicant," "License," "Test," and "Payment" with complicated relationships in DriverPass. Object modeling allows us to represent such relationships properly. For example, "Applicant" can be in a "one-to-many" relationship with "License" (one applicant can have many licenses during his or her life). Similarly, "License" can "need" a "Test" in certain situations.

• Encourages code reusability and maintainability. Object-oriented principles such as inheritance and polymorphism are central to object modeling. For instance, different "License" types (learner's permit license, full license) can inherit common attributes and behavior from an abstract class "License".

This reusability reduces code redundancy, minimizes development effort, and makes the system easy to modify and maintain. In modifications, base classes also map changes down to their derived classes automatically.

Encapsulation of data and behavior within objects encourages modularity since bugs are better and easier to isolate.

Object models may be implemented with code, and that improves model and code consistency.

**Disadvantages:**

• The process flows are less easily visualized by users using this method. UML sequence diagrams may be used to show some process flow but are typically used to show the interactions of objects for a particular use case, and not the overall systems process flow.

• Needs more object-oriented knowledge. Object modeling requires a good understanding of object-oriented principles such as classes, objects, inheritance, polymorphism, and encapsulation. This may be difficult for stakeholders without knowledge of these principles. Bridging the conceptual gap might take extra effort in effective communication between non-technical stakeholders and developers.