# CS 255 Model Application Short Paper

Dylan Cavazos

Dylan.Cavazos@snhu.edu

Southern New Hampshire University

## Process Model Application

Building a Process Model for the system DriverPass involves incorporating various techniques like Data Flow Diagrams. Data Flow Diagrams give you the ability to visualize the flow of data between different processes within a system and utilize four different symbols for these processes. These include data flows, data stores, processes, and sources/sinks, which all work together to build the fundamental structure of a data flow diagram for process models (Valacich, 2019, pp.205).

In this instance, an example of a data flow for DriverPass can be the student registration since this information is moved throughout the system to various stores. Another example is the lesson plans, which include details like the instructor, and the date and time of the lesson, which will be accessed and stored in various stores. User profiles can also be used as a data store, details from the user such as personal data, contact data, and login credentials are required to be held in the system for access by the user and the admin and instructor when needed.

Processes are important to Data Flow Diagrams since they represent how the flow of information is represented and by a function of ‘what’ that drives the flow of information. Processes for DriverPass include the user registering for a lesson, since the information is required to be captured by the system and accessed by an instructor for availability to plan a session. For the data sources/data sinks, examples include active users of the system like the students, instructors, secretary, and administrator. The instructor for example, acts as a source when they provide their availability and act as a sink when they garner lesson assignments for students.

In summary, utilizing Data Flow Diagrams for a system such as DriverPass involves incorporating the hierarchical nature of DFD’s and decomposing processes into lower-level diagrams (Valacich, 2019, pp.205). Overall, a context diagram can be implemented which can construct the overall process of DriverPass, followed by generating a level-o diagram capturing only the high-level processes of DriverPass like lesson scheduling for instructors and students. Finally, utilizing a process model in the design of DriverPass provides an overview of the higher-level structure of the various processes involved in the information flow between users and instructors.

## Object Model Application

To apply an object model for DriverPass, first it’s necessary to identify the key components like the objects and classes that will be implemented. Some examples for classes can be the “User”, “Instructor”, and “Lesson”. Each of these classes would have specific attributes inherit to their class. The “User” class would have attributes like username, password, and methods that include register or login. For the “Lesson” class some attributes that would be included are lesson ID, date, time, and location. Method examples include “scheduleLesson”, “viewStatus”, or “practiceTest”. As for the “Instructor” class I would inherit attributes from the “User” class but also include methods like “assignStudent”, “pickupLocation”, “dropoffLocation” and designate methods that are intrinsic to the “Instructor” class.

Object models also encompass aspects of entity-relationships between the classes and objects to signify how they are related. For instance, the “User” class could instantiate objects relating to the “Lesson” class since a single user may be completing more than one lesson. This representation of relationships can be modeled using a UML diagram to display the relationships between each class and object.

Incorporating aspects of Class inheritance is crucial to accurately capture the relationships between multiple entities. Another characteristic of object-orientation modeling is utilizing polymorphism as well. Allowing multiple classes to share similar methods like a “User” and “Instructor” class to share a “viewProfile” method would help portray the shared relationship often seen in object modeling.

Object modeling can be used to represent the relationships between multiple users through shared functionalities and common methods. Characteristics such as inheritance, abstraction, and polymorphism can be utilized to accurately capture these aspects between multiple classes like the “User”, “Instructor” and “Lesson” class.

## Process and Object Model Comparison

The advantages for process flow for DriverPass are that process models use techniques like Data Flow Diagrams which can accurately represent how data flows throughout a system. This makes it easier to understand how the overall system works. Another aspect is that process modeling allows for the creation of use-case diagrams, which help in determining user requirements and high-level system functionalities.

For Object modeling, the advantages are that object modeling utilizes an object-oriented approach using objects, classes, attributes, and methods which can help represent the components and behaviors of the system. Object modeling is also useful when representing the relationships between classes like “User” and “Lesson”, showing how one instantiates the other, or through inheritance, where classes can take on properties of other classes.

The disadvantages for process modeling, however, are that process modeling can make it difficult to manage very complex and intricate systems. Representing the process flows between each aspect of an intricate and large system can become difficult to follow and accurately capture. Process models also lack the necessary structure for representing behaviors and relationships within objects of a system, which is where object modeling picks up.

For object modeling, the disadvantages are due to having a steeper learning curve associated with modeling complex systems as understanding an object-oriented class structure can be difficult. When there are many classes to represent it may be difficult to accurately capture all of the relationships between each class and also show the inheritance and abstraction or polymorphism between each class.

## References

Valacich, J. S., & George, J. F. (2019). Modern Systems Analysis and Design (9th ed.). Pearson Education (US). https://mbsdirect.vitalsource.com/books/9780135172827