# CS 255 Model Application Short Paper

Julianne Takaya

Julianne.Takaya@snhu.edu

Southern New Hampshire University

## Process Model Application

A process model is a graphic representation of the work or actions, that are performed on data through in a given system. It is one of three major complementary views of an information system, which also includes logic and data models (Valacich & George, 2024). Process models are created iteratively in layers, starting with the most basic overview of the system and progressing into more and more detailed processes. A data flow diagram, or DFD, is the most common way to depict a process.

The first layer is a context diagram which shows which elements are outside of the system, and which are inside of it. For the DriverPass scenario, the elements outside of the system would include the Client, where client data originates, the Appointment Schedule database, where data about scheduled appointments are stored, the Lessons database, where the online lessons and practice tests are stored, and the DMV database, where the rules and regulations are checked. For the context diagram, the process would be DriverPass System. From there, I can break the DriverPass system into more specialized processes, such as Schedule Appointment, Take Practice Test, or Update Information. Once the second layer of the DFD is completed I would review it and decide if the processes need to be broken down into subprocesses through a process called functional decomposition (Valacich & George, 2024). This process would continue until no process can be logically broken down any further, at that point we have arrived at a primitive DFD, which is the most detailed form of the DFD. This level of detail can take several iterations to achieve depending on the complexity of the system being developed.

## Object Model Application

In terms of computer science and software engineering, an object model is a model that uses object-oriented techniques. Object-oriented programming involves modeling real-world objects as software objects. In this approach we would take an object, such as the Customer from the DriverPass system, and assign data that describes the Customer, such as customer name, address, phone number, and email, and store it within the Customer object. These objects are often depicted in the way of a Unified Modeling Language (UML) diagram. This diagram shows the behavior and structure of a system. The most widely used UML diagram is the class diagram (Unified Modeling Language (UML) Diagrams, 2024). A class diagram shows a high-level overview of a system’s design and helps communicate the structure of the software. In the case of the DriverPass system, it would be reasonable to create a Customer class to store the data about customers as attributes.

The Customer class would also contain the methods or functions of the Customer class. These functions define the behaviors of the class. In this diagram, the attributes of the Customer class would include Name, Address, Phone number, and Email. The methods would include Update Name or Update Phone Number, to allow for changes to be made, but also Retrieve Name or Retrieve Email, to allow other classes to use this data if needed. Class diagrams are a great way to depict the functionalities and relationships within the system. For instance, in the system we would need to create objects for Customers and Administrators. We could model this by creating a parent class called User, which would contain all of the shared attributes between Customers and Administrators such as Name and Phone number. Then, the child class of Administrator would have different attributes and methods that are separate from the attributes and methods of the Customer class.

## Process and Object Model Comparison

A process model can be an important tool for analysists to use as they can show discrepancies within a system. Issues such as redundant data flows, data that is collected but never used, or data that is updated redundantly are all issues that can be shown in a DFD. Seeing these issues in the DFD can help analysts redesign the system to be more efficient in the future. However, there is a risk of over-analyzing a process to the point of redundancy. It is difficult to know when to stop breaking processes down.

A complete UML diagram would show all the necessary classes and their relationships with each other. From this model we can see if the proposed system would meet the client’s expectations of functionality. If the system does not do what the client wants done, it is not a complete system. While object models have many advantages, there are some considerations, as well. Object models can be time-intensive to create and may become quickly outdated if project requirements change.

Both process models such as DFDs and object models such as UMLs can be great tools for analysts to use as an overview of a system. One is not better than the other, rather, each diagram has a distinct purpose. Process models show the dynamic system, or the data in motion where object models show the static state of the data. Analysts use both models to design and improve systems.

## References

Unified Modeling Language (UML) Diagrams. (2024, June 25). From Geeks for Geeks: https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/

Valacich, J., & George, J. F. (2024). Modern systems analysis and design, 10th edition. Pearson.