# CS 255 Model Application Short Paper

Tyeika Willis

Tyeika.Willis@snhu.edu

Southern New Hampshire University

## Process Model Application

To apply a process model to the DriverPass project, we would focus on capturing and documenting the sequence of tasks and activities involved in the system’s operation. The goal would be to provide a step-by-step flow that describes how users interact with the system and how the system fulfills the business objectives. Process modeling would help define workflows such as how students schedule lessons, complete online practice tests, and track their progress. Key tasks like payment processing, lesson booking, and test result analysis would be mapped out clearly, showcasing the flow of data and actions across various modules.

For DriverPass, a process model would also define interactions between different user groups, such as students, instructors, and administrators. For instance, the process of booking a driving lesson could include steps like selecting an instructor, choosing a time slot, confirming payment, and generating a booking confirmation. Each step would have associated inputs, outputs, and rules, ensuring all possible scenarios are accounted for. This approach provides clarity and ensures a detailed understanding of how the system will function in real-world scenarios.

## Object Model Application

An object model, on the other hand, organizes the system design around real-world entities and their interactions. For DriverPass, the object model would identify key objects such as Student, Instructor, Lesson, Test, and Payment. Each object would have attributes and methods relevant to its role in the system. For example, the Student object might include attributes like name, email, and test scores, along with methods such as scheduleLesson() or viewProgress().

Using this model, we would create a modular system where each object encapsulates its data and behaviors. This makes it easier to extend or modify the system as DriverPass evolves. For example, if the company decides to add group lessons in the future, we could simply extend the Lesson object to include a group mode without impacting the rest of the system. Additionally, objects interact with each other to complete tasks. For instance, when a student books a Lesson, the Lesson object updates its schedule, and the instructor object adjusts their availability.

The object model’s focus on reusability and modularity ensures a scalable and maintainable system design. It is particularly suited for handling the complexity of DriverPass’s requirements, such as integrating online practice tests with in-person lesson scheduling.

## Process and Object Model Comparison

Both process and object models have their strengths and limitations when applied to the DriverPass scenario. The process model excels at mapping out workflows and providing a clear understanding of how the system operates. It ensures that every step is accounted for and helps identify potential bottlenecks or inefficiencies. However, it can become rigid, making it harder to adapt to changes or add new features without reworking the entire flow.

The object model, by contrast, offers flexibility and reusability. It allows for easier updates and modifications, as objects are self-contained and interact in defined ways. This makes it a better choice for complex systems like DriverPass, where multiple user roles and interactions need to be managed. However, object modeling can be more abstract, and without detailed planning, it may overlook specific workflow nuances that the process model captures effectively.

For DriverPass, a combined approach could be ideal. The process model can be used to define high-level workflows, ensuring all tasks and interactions are understood. The object model can then implement these workflows in a modular and scalable way, ensuring the system can adapt to future changes.