# CS 255 Model Application Short Paper

Cassim Mabirizi

[Cassim.Mabirizi@snhu.com](mailto:Cassim.Mabirizi@snhu.com)

Southern New Hampshire University

## Process Model Application

The Process Model focuses on defining sequential workflows and activities to ensure all steps in a system are logically and methodically completed. For the DriverPass scenario, this model could be applied by outlining the flow of user interactions, such as scheduling lessons, attending practice sessions, and completing mock exams. Each step would follow a predefined sequence, ensuring that all user activities are carried out systematically.

For example, the process would start with a user registering an account, followed by lesson scheduling, lesson completion, practice exam attempts, and finally, mock test grading. This structured approach ensures that each activity is completed in a logical order, reducing potential errors or missed steps in the system.

**Advantages**

* The Process Model is straightforward, making it easy to implement and understand for both users and developers.
* It is especially well-suited for simple, linear workflows like test grading or session tracking, where tasks follow a predictable sequence.

**Challenges**

* This model is rigid, which can make it less adaptable to changes or unexpected user behaviours.
* For dynamic and interactive features like rescheduling lessons or offering tailored practice recommendations, the Process Model may be inefficient.

## Object Model Application

The Object Model emphasizes defining the system in terms of objects, their attributes, and their interactions. In the context of DriverPass, this would involve identifying key objects such as User, Lesson, Exam, and Instructor. Each object would have its own attributes and methods. For instance:

* **User:** Attributes include username, email, and registeredLessons; methods include ScheduleLesson() and ViewProgress().
* **Lesson:** Attributes include date, time, and instructor; methods include AssignInstructor() and Reschedule().
* **Exam:** Attributes include questions, score, and status; methods include GradeExam() and GenerateReport().

By focusing on these entities and their interactions, the Object Model allows for a flexible and reusable design. Developers can extend functionality easily by adding new objects or enhancing existing ones.

**Advantages**

* This model supports reusability, making it easier to scale the application as new features or objects are introduced.
* It provides a clear structure for dynamic, user-focused interactions, such as rescheduling lessons or providing personalized feedback based on exam performance.

**Challenges**

* The Object Model requires more initial effort to define and design objects, which can be time-consuming.
* It may also have a steeper learning curve for team members unfamiliar with object-oriented principles.

## Process and Object Model Comparison

The **Process Model** is highly effective for linear and predictable workflows, making it well-suited for tasks in DriverPass such as mock exam grading or straightforward lesson scheduling. Its simplicity allows for ease of implementation and understanding by both developers and end users. However, the rigid structure of the Process Model can pose challenges when adapting to changes or accommodating complex, dynamic user interactions.

In contrast, the **Object Model** provides notable advantages in terms of scalability and flexibility. Its modular design is ideal for handling intricate tasks, such as managing dynamic user interactions, supporting personalized feedback, and integrating additional features over time. This adaptability ensures that DriverPass can grow with user demands and technological advancements. However, the Object Model comes with its own challenges, including a steeper learning curve for teams unfamiliar with object-oriented principles and a greater initial time investment to define and design the system’s objects effectively.

**Recommendation**  
Considering DriverPass's requirements, such as dynamic lesson scheduling, personalized user interactions, and the potential need for future scalability the Object Model emerges as the superior choice. Despite the higher initial effort required, its flexibility and capacity to handle complexity make it better suited to supporting the system's long-term success.