# CS 255 Model Application Short Paper

Rosalinda Castaneda

Rosalinda.castaneda@snhu.edu

Southern New Hampshire University

## Process Model Application

In computer science, a process model represents a structured function within a system’s architecture. It serves to clarify how specific tasks are carried out, outlining the system’s operations and objectives. These models are generally built in layers and help visualize the system’s final outcome. When constructing a process model, it’s important to aim for three primary purposes: describing the system, prescribing what should happen, and explaining how the system functions.

Process models are commonly categorized by four key attributes: coverage, alignment, granularity, and flexibility. A popular tool used in developing such models is the Data Flow Diagram (DFD). Other modeling tools include UML diagrams, flowcharts, and role activity diagrams. When working with a DFD, four main principles should be followed: ensuring completeness, consistency, attention to timing, and allowance for iteration.

Applying this to the DriverPass case, the first step would be to analyze the client's requirements. According to the interview, the client wants the system to support student preparation for driving tests and provide multiple session package options. These objectives shape the scope of the model.

The primary actors in this scenario are instructors, students (users), and system administrators. Instructors should be able to monitor and record student session details, such as lesson duration, time slots, and feedback. Students, in turn, must be able to register accounts using valid credentials, choose among available packages, take practice exams, track their performance, and access additional learning materials offered.

System administrators play a crucial backend role. Their responsibilities include managing user access, deactivating outdated or inactive session packages, and handling password resets. These functions ensure the platform runs smoothly and securely for all users.

## Object Model Application

An object model represents a system using object-oriented principles. These models are typically visual tools used to illustrate how objects interact within a system. They rely on object-oriented programming concepts to structure the system into manageable components. Well-known examples of object models include the Document Object Model (DOM) and the Component Object Model (COM).

When designing an object model, several elements must be incorporated: interfaces, object references, actions, and exceptions. These components ensure the model accurately captures how different parts of the system communicate and behave.

To apply the object modeling approach to the DriverPass case, we must identify and define key object-oriented elements: abstraction, encapsulation, modularity, and hierarchy. These principles guide the organization and structure of the system’s components.

With this foundation, we would begin by distinguishing between tangible and intangible entities within DriverPass, for example, users, lesson sessions, and exam results as objects. We would also define how these objects relate to one another. Understanding these relationships is essential to meeting the functional goals of DriverPass, such as allowing students to access resources, instructors to track progress, and administrators to manage user roles and data.

## Process and Object Model Comparison

Both process models and object models offer unique advantages, and choosing the best fit for DriverPass requires a thoughtful evaluation. Process models are particularly effective for illustrating the layered architecture of a system. They help stakeholders visualize how different parts of the system interact and provide a clearer picture of the expected final product. For a project like DriverPass, using a process model could highlight potential problem areas early on, allowing for necessary revisions or optimizations. In essence, a process model helps align the system design with the business goals.

However, a key drawback of process modeling is the risk of overanalyzing. Given that DriverPass is operating on a tight timeline, as seen in the Gantt chart, getting too caught up in perfecting the process model could delay development. This kind of inefficiency could prove costly.

On the other hand, object models also offer visual insight, but they focus more on defining system components such as objects, interfaces, actions, and exceptions. Object modeling tends to be faster and more agile, which would be beneficial for DriverPass given the limited time allocated for planning. The trade-off, however, is that object models can lack a clear structure if not designed carefully, and they may require developers with solid experience in object-oriented design to implement effectively.

While both approaches have merit and could complement one another, object modeling seems like the more practical choice for DriverPass. It offers a quicker path to development and aligns better with the project’s time constraints.