# CS 255 Model Application Short Paper

Michael Watkins

michael.watkins5@snhu.edu

Southern New Hampshire University

## Process Model Application

## Identify Core Processes: Determine the key processes that are essential for the DriverPass system. These might include scheduling driving lessons, managing instructor availability, processing payments, and tracking student progress.

## Define Process Flows: For each identified process, outline the sequence of steps involved. For instance, the process for scheduling a lesson might start with a student selecting a preferred time slot, the system checking instructor availability, and then confirming the booking.

## Establish Inputs and Outputs: Identify what inputs are required for each process and what outputs or results they produce. For example, the input for the lesson scheduling process could be the student’s preferred times, and the output would be a confirmed appointment.

## Map Interactions Between Processes: Show how different processes interact with each other. For instance, how the payment process is initiated once a lesson is scheduled, or how instructor availability updates affect the scheduling process.

## Incorporate Decision Points: Include decision-making steps within the processes. For example, how the system decides what to do if a preferred lesson slot is not available, or how it handles rescheduling requests.

## Detail Exception Handling: Plan for exceptions or unusual scenarios. For instance, how the system responds to a payment failure, or what happens if an instructor suddenly becomes unavailable.

## Optimize for Efficiency: Analyze each process for potential inefficiencies or bottlenecks and refine them for smoother operation. This could involve streamlining steps, automating certain tasks, or improving the user interface for easier navigation.

## Ensure Alignment with User Needs: Continuously refer back to the business requirements and user needs to ensure that the process model accurately reflects what is needed for a successful DriverPass system.

## Object Model Application

## Identify Key Objects: Start by identifying the main objects involved in the DriverPass system. These might include Student, Instructor, Lesson, Schedule, Vehicle, Payment, and Account.

## Define Attributes for Each Object: For each object, determine its attributes. For example, the Student object might have attributes like studentID, name, address, email, and lessonHistory. The Instructor object could have instructorID, name, schedule, and vehicleAssigned.

## Define Methods for Each Object: Assign relevant methods (actions or functions) to each object. For instance, the Lesson object might have methods like scheduleLesson(), cancelLesson(), and updateLessonDetails(). The Payment object could have methods like processPayment(), refund(), and verifyTransaction().

## Establish Relationships Between Objects: Determine how these objects interact with each other. For example, a Student is associated with multiple Lessons, and each Lesson is taught by an Instructor. There could be a one-to-many relationship between Instructor and Lesson.

## Incorporate Inheritance Where Applicable: If there are generalizations in the system, use inheritance. For example, if there are different types of users like Student, Instructor, and Admin, they could all inherit from a general User class that contains common attributes like userID, username, password, and contactInfo.

## Consider Aggregation and Composition: Identify where aggregation or composition relationships exist. For instance, a Schedule might be a composition of Lessons, implying that if the Schedule object is destroyed, so are the Lessons within it.

## Model Behavioral Aspects: Consider how objects change state. For example, when a Lesson is completed, its status might change, which could trigger updates in the Student’s lesson history and the Instructor’s schedule.

## Review for Completeness and Consistency: Ensure that the object model captures all aspects of the DriverPass system as per the requirements. Check for consistency in naming conventions and relationships.

## Process and Object Model Comparison

For DriverPass, each modeling approach, has unique advantages and disadvantages.

**Process Model Advantages:**

1. **Clarity in Workflow**: It clearly outlines the step-by-step workflow of the DriverPass system, making it easier to understand and analyze the sequence of operations, from scheduling lessons to processing payments.
2. **Identification of Bottlenecks**: Helps in identifying potential bottlenecks or inefficiencies in the system, which is crucial for optimizing the workflow and improving user experience.

**Process Model Disadvantages:**

1. **Lack of Structural Detail**: It doesn't provide much insight into the system's underlying structure, such as how data is organized or how different components are related.
2. **Limited Behavioral Insight**: Process models don't typically capture the dynamic behavior of the system, like how objects interact or change state over time.

**Object Model Advantages:**

1. **Detailed System Structure**: Offers a detailed view of the system's structure, showing how different entities (like students, instructors, lessons) are related and interact.
2. **Facilitates Modular Design**: Supports a modular approach to system design, which can be beneficial for maintenance and scalability.

**Object Model Disadvantages:**

1. **Complexity**: Can become complex, especially for large systems with many entities and relationships, making it harder to grasp the overall system functionality.
2. **Doesn’t Illustrate Process Flow**: Doesn’t explicitly show the sequence of actions or how tasks are carried out within the system.

**Behavioral Model Advantages:**

1. **Dynamic System Behavior**: Captures the dynamic aspects of the system, such as state changes, events, and interactions, providing a deeper understanding of how the system operates in real-time.
2. **Useful for Simulation**: Can be used to simulate the system’s behavior, which is valuable for testing and validating system functionality.

**Behavioral Model Disadvantages:**

1. **Can Be Abstract**: Sometimes, behavioral models can be abstract and not straightforward for stakeholders who are not well-versed in technical details.
2. **Integration Complexity**: Integrating behavioral models with other aspects of system design (like the database structure) can be challenging.

## References

Valacich, J. S., & George, J. F. (2017). *Modern Systems Analysis and Design* (8th ed.). Pearson.