# CS 255 Model Application Short Paper

Jazzlynn Gonzalez

jazzlynn.gonzalez@snhu.edu

Southern New Hampshire University

## Process Model Application

A process model focuses on how information flows through a system and how users interact with it at each step. In the DriverPass, the model would help outline the sequence of actions students, instructors, and staff take when using the system. For example, the process would start when a student registers for an account, continues as they schedule driving lessons, take online practice tests, and ends when they review their progress or instructors feedback. By mapping these steps, the team can understand how each part of the system connects and identify where problems might occur before development begins.

Using a process model in system design helps ensure that all user activities are clearly defined and logically ordered. According to Shelly and Rosenblatt (2012), process modeling helps analysts describe what the system does, not how it will be implemented, which allows for better planning and communication between developers and clients. In the case of DriverPass, the process model would illustrate workflows such as lesson scheduling, payment handling, and data sharing between users and the database. For instance, when a student books a lesson, the system should send the request to the database, verify the instructor’s availability, and return a confirmation message.

Process models are also useful for identifying dependencies and redundancies. Satzinger, Jackson, and Burd (2016) explain that by documenting data flows between users and the system, analysts can make sure all necessary information is captured, processed, and stored correctly. For DriverPass, this could prevent scheduling conflicts, data duplication, or missing progress reports. Overall, applying a process model ensures that the DriverPass system supports a smooth, efficient, and reliable experience for students and staff by making each process clear and easy to follow.

## Object Model Application

An object model focuses on identifying the main parts of a system called objects. Each object represents something real within the system, such as a person, place or event, and contains both data and actions it can perform. In the DriverPass, examples of key objects include Student, Instructor, Lesson, PracticeTest, and Administrator. Each of these objects would have unique attributes and behaviors that define how they interact with one another.

For example, the Student object would contain information such as the student’s name, contact details, progress scores, and schedule lessons. It would also have behaviors such as registerAccount(), scheduleLesson(), and takeTest(). The Lesson object would store lesson details like date, time, car, instructor, and lesson notes. While the instructor object would include data such as instructor ID, availability, and assigned students. These objects would connect through relationships, such as one to many link between Instructor and Lesson, to show how information is shared across the system.

According to Satzinger, Jackson, and Burd (2016), object modeling helps developers organize data and behavior in a way that mirrors real world interactions which makes the system easier to understand and maintain. By applying this model to DriverPass, the design team can create resuable classes that can be updated without changing the entire system. For example, if DriverPass later adds a new TrainingPackage object for special lessons, it could be added without affecting other parts of the design.

Shelly and Rosenblatt (2012) explain that object modeling is especially useful in systems that involve multiple user roles because it clearly defines how each user interacts with the system and what actions they can perform. In the DriverPass, the Student object interacts with the PracticeTest object to record scores. While the Instructor object updates lesson results for each student. This clear structure ensures consistency and reduces duplication across the database.

Overall, using an object model helps the DriverPass team create a system that is organized, flexible, and scalable. It makes future updates easier and ensures that relationship between users, lessons, and training materials remain well-structured and easy to manage.

## Process and Object Model Comparison

Both the process model and the object model provide valuable ways to design and understand the DriverPass system, but they each have different strengths and weaknesses depending on the goal of the project. The process model is helpful because it focuses on how information and activities move through the system. It shows the step by step flow of data which helps identify what happens first, next, and last. This makes it useful for understanding how students register, schedule lessons, take practice tests, and view their progress. Shelly and Rosenblatt (2012) explain that process models are especially useful in early development because they help visualize workflows and ensure that every action is accounted for before building the system. For DriverPass, the process model would help map out tasks such as scheduling lessons, handling payments, and updating practice exam results. However, a disadvantage of the process model is that is can be difficult to update if the system changes. Satzinger, Jackson, and Burd (2016) note that process models can become complex when multiple users and processes interact. For example, if DriverPass adds new user types or lesson packages, the process flow diagrams would need major updates. This makes them less flexible for long-term maintenance once the system starts evolving.

One the other hand, the object model focuses on objects of the system and how they relate to one another. If offers more flexible and realistic way to organize data and system behavior. For DriverPass, using an object model makes is easier to represent key parts of the system such as Student, Instructor, Lesson, and PracticeTest. According to Satzinger, Jackson, and Burd (2016), object models simplify software development because the same objects can be reused or modified without changing the entire system. For example, if DriverPass wants to add a new “refresher course” option, developers could simply create a new object class without redesigning all the workflows. The disadvantage of the object model is that it can sometimes be harder for non-technical stakeholders to understand. Shelly and Rosenblatt (2012) mention that object models require careful planning and programming knowledge to correctly define relationships and behaviors between objects. In DriverPass, the project team may need to spend extra time explaining how classes and objects work to ensure that the client understands how the design supports their goals.

Overall, both models work best when used together. The process model helps visualize the flow of work and ensures no steps are missed while the object model helps structure the system’s data and behaviors for long-term growth. By combining the two, the DriverPass team can build as system that is both clear in function and flexible in design.

## References

Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2016). *Systems analysis and design in a changing world* (7th ed.). Boston, MA: Cengage Learning.

Shelly, G. B., & Rosenblatt, H. J. (2012). *Systems analysis and design* (9th ed.). Boston, MA: Cengage Learning.