**CS-255 System Analysis And Design**

**Module 5**

**Project One**

**Model Application Short Paper**

**Submitted By:**

HAMNA KHALID

ID: 2902671

[hamna.khalid@snhu.edu](mailto:hamna.khalid@snhu.edu)

**Submitted To:**

PROFESSOR PARUL HIRPARA

p.hirpara@snhu.edu

Southern New Hampshire University

Thursday, October 02, 2025

**Model Application Short Paper**

Designing effective systems requires carefully selecting the appropriate modeling approaches. For the DriverPass project, which aims to provide students with online training, practice exams, and on-the-road lessons, two main approaches stand out: the process model and the object model. Each offers unique strengths in representing system requirements and structure, and comparing them helps determine which is best suited for different parts of the project. This paper will apply both models to the DriverPass scenario, discuss their advantages and disadvantages, and consider how they can be combined to support a complete system design.

# **Process Model Application**

The process model focuses on how information flows through the system and what actions must be completed to achieve a task. In the case of DriverPass, a process model can map out student interactions, including registration**, booking lessons, taking online tests, and tracking progress**. It would also include staff tasks such as **scheduling appointments, resetting accounts, and generating reports**.

For example, a **data flow diagram (DFD)** could show how customer input (like personal information, payment details, or package selection) flows into the system and produces outputs such as confirmation messages, schedules, and reports. This approach helps visualize step-by-step activities, such as a customer booking a lesson, which requires interactions between the customer, secretary, trainer, and database.

The main benefit of the process model for DriverPass is that it clearly shows **dependencies and sequences of tasks**, making it easier to identify inefficiencies, bottlenecks, or missing steps in workflows. For example, ensuring that a reservation cannot be confirmed without payment processing would be explicitly represented. However, a limitation is that process models focus only on **how** things happen, not on the structure of the system or its long-term adaptability.

# **Object Model Application**

The object model, by contrast, focuses on the **system's structure**, defining classes, attributes, methods, and relationships. For DriverPass, the object model would define classes such as **Customer, Account, Reservation, Vehicle, Trainer, Package, IT Officer, and Administrator**. Each class encapsulates data and behaviors; for instance, a Customer object might include attributes like *name* and *address*, and methods like *makeReservation()* or *resetPassword()*.

This approach is beneficial because it organizes the system into modular, reusable components. For example, the **Reservation object** can consistently handle booking details regardless of whether the appointment was created by a customer online or by the secretary in person. Similarly, the **Package object** can define multiple training options (6, 8, or 12 hours, with optional DMV prep) while remaining flexible for future changes.

The strength of the object model is its alignment with **Object-Oriented Programming** principles like encapsulation and polymorphism, making it directly useful for developers during implementation. However, a limitation is that it does not show the **flow of activities over time**, which means understanding the order of processes requires complementary models like activity or sequence diagrams.

# **Process And Object Model Comparison**

Both models are valuable, but they emphasize different perspectives of the same system. The **process model** excels at showing workflows and how data moves through the system, making it easier to capture user requirements and business rules. For DriverPass, it ensures the system supports every task needed by customers, staff, and administrators. The drawback is that it does not provide a strong foundation for coding or long-term scalability.

The **object model**, on the other hand, excels at defining the **building blocks of the system**, which are essential for developers. It ensures that each class has a clear role and that relationships (such as a Customer having multiple Reservations) are well-defined. The drawback is that it does not capture the sequence of steps as clearly as a process model.

Together, these two approaches complement each other. The process model shows the **“how”** of the workflows, while the object model shows the **“what”** of the system’s structure. For DriverPass, combining them ensures the system is both well-designed for developers and user-friendly for customers and staff.

# **Conclusion**

The DriverPass project needs a comprehensive system that handles both customer-facing tasks, such as lesson scheduling and online practice tests, as well as administrative functions like account management and compliance reporting. The process model clarifies workflows and task sequences, while the object model offers a structured blueprint for coding and future scalability. By utilizing both, DriverPass can develop a reliable, secure, and flexible system that addresses the needs of students, trainers, and administrators alike.

# **References**

* Dennis, A., Wixom, B. H., & Tegarden, D. (2015). *Systems analysis and design with UML* (4th ed.). Wiley.
* Valacich, J. S., & George, J. F. (2020). *Modern systems analysis and design* (9th ed.). Pearson.
* Southern New Hampshire University. (2025). *CS-255 Systems Analysis and Design course materials*.
* Khalid, H. (2025). *CS-255 Module Three Assignment: Process Model Evaluation* [Unpublished student paper]. Southern New Hampshire University.
* Khalid, H. (2025). *CS-255 Module Four Assignment: Object Model Evaluation* [Unpublished student paper]. Southern New Hampshire University.