CS 255 Model Application Short Paper

Megan Mitchell

Megan.mitchell6@snhu.edu

Southern New Hampshire University

Process Model Application

For the DriverPass project, I would apply a process modeling approach to clearly outline the step-by-step workflows and data movements that occur in the system. My main goal with this model would be to understand how each process interacts with users, data, and other components, so that I can ensure nothing is missed when the system is developed. As shown in the diagram above, a process model focuses on how tasks are completed, as well as how operations are completed. This is especially important for a system like DriverPass, which relies heavily on scheduling, user management, and secure data handling.

In the DriverPass scenario, there are several core processes that need to be represented: account creation, lesson package selection, online payment, scheduling driving lessons, modifying or canceling appointments, completing online practice exams, and tracking progress. Each of these processes involves multiple users including students, instructors, secretaries, and administrators. I would use data flow diagrams (DFDs) to visualize how data moves between external entities like the DMV, internal processes like scheduling, and data stores like the user database. This allows me to trace where every piece of information originates and where it ends up.

For example, in the 'Schedule Driving Lesson' process, the student logs in and requests an available time slot. A driver and car availability check is conducted through this system, which ensures the student's active lesson package is validated, and bookings are confirmed. There are several steps that would need to be captured in a process model. Including alternate flows would help describe what happens if there's a conflict, like a double-booking, expired

package, or a payment failure. By identifying these early through process modeling, I can make sure the system handles errors smoothly.

Another major benefit of process modeling is that it highlights dependencies between departments or users. For instance, the secretary must be able to view or edit a student's appointment if they call the office. Therefore, the scheduling process must be capable of handling both manual and online changes without causing any data conflicts as a result. The admin must be able to monitor activity logs and reset passwords, which adds additional security processes into the model, such as authorizations and authentications, in order to maintain a secure model. It is important that I document these workflows first so that I can lay a solid foundation for the subsequent phases of technical design.

Overall, I would use process modeling to help visualize every functional area of DriverPass before coding begins. This method ensures that all business rules, timing requirements, and dependencies are defined, reducing the risk of missing important system behaviors later in development.

Object Model Application

When I apply an object modeling approach to the DriverPass project, my focus shifts from workflows to the actual entities, or objects, that exist within the system and how they interact with each other. Object modeling is used to define the structure of the system, what types

of data will exist, what behaviors those data have, and how they relate. This helps me ensure the system is logically organized, reusable, and easy to maintain.

The first step I would take is identifying the main classes in DriverPass. These include User, Student, Instructor, Admin, Secretary, Vehicle, Lesson, Reservation, Package, OnlineTest, TestAttempt, DMVUpdate, and AuditLog. Each of these classes has specific attributes and behaviors. The Student object, for example, might include attributes like the name, address, phone number, package type, progress report, as well as behaviors such as scheduling a lesson or taking a practice test, for example. Objects associated with vehicles would contain details such as car IDs, models, and maintenance statuses, as well as information about the instructor who owns the vehicle. The relationships between these classes show how the system's parts connect to one another.

In addition to attributes, each class has methods that define how objects behave. For example, the Student class would have methods like scheduleLesson() and cancelLesson(), while the Admin class might include resetPassword() or disablePackage(). These methods correspond directly to the functional requirements outlined in the DriverPass interview. By defining them early, I can make sure that each role's responsibilities are properly encapsulated in the system's design.

Another important aspect of the object model is its use of inheritance and polymorphism. It is for instance the User class that serves as the parent class for all users that has all the basic information and methods shared between them. Students, Instructors, Admins, and Secretaries

would all inherit from User, but each of these classes would have unique methods suited to their particular roles. This approach reduces redundancy in the system design and improves scalability as DriverPass grows or adds new types of users in the future.

Finally, object modeling makes it easier to maintain and expand the system. If the DMV updates testing standards or new features are introduced, I can simply modify or extend the relevant classes without redesigning the entire architecture. By using this modular approach, the system is not only more efficient, but also more collaborative among developers working on different components.

Process and Object Model Comparison

Both the process and object models are valuable for designing a system like DriverPass, but they serve different purposes. The process model focuses on the flow of information, how tasks are completed from start to finish, while the object model focuses on the structure of the system, what data and components make up the system. Together, they provide a complete picture of how DriverPass should operate.

The advantages of the process model include its ability to clearly show workflows, dependencies, and data movement. It is especially useful when explaining the system to non-technical stakeholders like Liam, the owner of DriverPass, or the secretaries who will use it daily. This model ensures that business logic is correctly understood before technical

development begins. However, its disadvantage, is that it doesn't show how data is stored or how individual system components interact internally.

The advantages of the object model, include its reusability, scalability, and the way it simplifies system maintenance. It allows developers to organize code around real-world entities, which makes it easier to extend the system as new requirements arise. Its disadvantage is that it can be more abstract and harder for stakeholders to interpret, especially when they are more focused on workflows than data structures.

In my opinion, the best approach for the DriverPass system would be to use a hybrid method that combines both models. The first step I would take would be to create a process model to map out all workflows, ensuring that all business requirements have been fully understood. Having done that, I would then use the object model to define how each process within the system is linked to the data and objects that belong to it. Combining these three factors ensures that the information is accurate, clear, and flexible over the long term.

By applying both methods together, I can help DriverPass achieve a system that not only meets current needs, such as lesson scheduling, testing, and data security, but also supports future updates as technology and DMV requirements evolve. Using both models also aligns with the principles of system development taught in this course, emphasizing structured design, iterative improvement, and client collaboration throughout the process.