# CS 255 Model Application Short Paper

Jacob Rethmeier

Jacob.rethmeier@snhu.edu

Southern New Hampshire University

## Process Model Application

In the context of designing a system for DriverPass, a process model serves as a vital framework to understand and streamline the development process. Process models offer a structured approach to organizing tasks, defining workflows, and managing resources effectively. In the case of DriverPass, which aims to take advantage of an empty market and provide a system for training student drivers, the application of a process model can significantly enhance the efficiency and success of the project. The two main process models applicable to the DriverPass scenario are the Waterfall model and the Agile model. The Waterfall model presents a sequential approach to software development, where each phase flows in a linear manner, and progress is measured by completing one phase before moving onto the next. The Agile model emphasizes flexibility, collaboration, and iterative development. It involves breaking down the project into smaller, manageable tasks and delivering incremental improvements based on client feedback.

Firstly, in the Requirement Analysis phase, the team gathers and documents all client requirements comprehensively. This includes identifying key functionalities such as online classes, practice tests, appointment scheduling, and behind-the-wheel training. Subsequently, in the System Design phase, a detailed system architecture is developed based on the gathered requirements. This involves designing system components such as the online classes platform, web server/host, appointment scheduling system, database management, and payment system integration.

Following the System Design phase, the Implementation phase begins, where the development process is initiated based on the finalized design. Each component of the system is developed and integrated, ensuring compatibility and functionality. After Implementation, thorough testing is conducted in the Testing phase. This includes unit testing, integration testing, and system testing to identify and rectify any issues or bugs.

Upon successful testing, the Deployment phase ensues, where the developed system is deployed after thorough configuration and setup of the web server/host, database, and security measures. Finally, in the Maintenance phase, ongoing support and maintenance are provided post-deployment. This includes addressing any user issues, performing updates as required, and ensuring the system remains operational and up-to-date. Client Demos and Feedback play a crucial role in gathering client input and prioritizing tasks, while Retrospective Meetings provide opportunities to reflect on project progress and refine processes.

## Object Model Application

In the design of a system like DriverPass, an object model serves as a foundational element to represent the structure and behavior of the system's components. An object model provides a visual representation of how data is organized and manipulated within the system, offering insights into the relationships between different entities and their interactions. In the context of DriverPass, the application of an object model is crucial for defining the system's architecture and facilitating effective communication among development teams.

One of the primary components of an object model is the class diagram, which illustrates the structure of the system by representing classes, attributes, methods, and relationships. For DriverPass, a class diagram can be created to represent various entities within the system. These include the User class, representing different types of users such as students, instructors, and administrators, each with attributes like user ID, username, password, and role. Additionally, classes like Course, Appointment, Payment, and Quiz encapsulate information and functionality related to courses, appointment scheduling, payment transactions, and quiz management, respectively. Another aspect of the object model is the object interaction diagram, which shows the interactions between objects within the system over time. Object interaction diagrams, such as sequence diagrams or communication diagrams, can be particularly useful for illustrating how different system components collaborate to fulfill user requests or perform system operations. For example, a sequence diagram can show the sequence of interactions between user objects, course objects, and payment objects during the enrollment process, providing a visual representation of how a user selects a course, makes a payment, and is enrolled in the course.

Both process and object models have their benefits in systems analysis and design, for object models first and foremost it would be to provide clarity and visualization, making it easier for developers to understand and communicate system design. By representing system components as objects and their relationships, object models promote modularity and reusability of code, enhancing maintainability and scalability. Object-oriented principles such as abstraction and encapsulation help manage system complexity by hiding internal implementation details and exposing only relevant interfaces. However, object models also come with certain disadvantages. The complexity of object models can increase, particularly in large-scale systems with numerous classes and interactions, making them challenging to manage and comprehend. Implementing an object-oriented design may introduce additional overhead in terms of memory and processing resources, particularly in systems with frequent object instantiation and manipulation. Additionally, object-oriented design principles may have a steep learning curve for developers transitioning from procedural or functional programming paradigms, requiring time and effort to master.

## Process and Object Model Comparison

In the design and development for the DriverPass system, the choice between process and object models is a critical decision that significantly influences project outcomes. Each model offers unique advantages but also has their own unique disadvantages. For example, when looking at the Waterfall model, it focuses on a structured and sequential progression through the analysis and design. The structured nature of the Waterfall model provides effective planning and management of the development process. However, its rigidity can become a limitation when faced with changes or uncertainties in project requirements, potentially impacting project timelines and costs. In contrast, the object model, rooted in object-oriented principles, offers a more flexible and adaptable approach to system design. Object-oriented design enables modularization and organization of system components in a manner that mirrors real-world entities and their interactions. This modular approach promotes code reusability, maintainability, and scalability, facilitating easier adaptation to changing requirements or new features. However, developing an object model is a far more complex process and is visually more difficult to understand. Additionally, managing relationships between objects and ensuring consistency and coherence can become increasingly complex as systems grow.

In considering the application of process and object models to the DriverPass scenario, the object model's flexibility and adaptability make it well-suited for projects that require iterative development and frequent changes. Another consideration is the potential size and outreach this system could have so the adaptability of object models seems more appropriate for this situation.

Top of Form