# CS 255 Model Application Short Paper

Rafael V. Canseco

Rafael.Canseco@SNHU.edu

Southern New Hampshire University

## Process Model Application

In computer science, a process model is a fundamental element within a broader system model. These process models serve to elucidate the functions within a model, effectively explaining what the model accomplishes within a given system. Typically, these models are structured in layers and provide a conceptual overview of the envisioned final product. In the realm of designing process models, three primary objectives come into play: descriptiveness, prescriptiveness, and explanatory capacity.

In the process of creating process models, four key classifications are commonly considered: coverage, alignment, granularity, and flexibility. Notably, Data Flow Diagrams (DFDs) stand out as a well-established example used for crafting process models. Additionally, other modeling techniques, such as Unified Modeling Language (UML) diagrams, flowcharts, and role-based activities, find applications in this domain.

When it comes to designing DFDs, adherence to four fundamental guidelines is essential: ensuring completeness, maintaining consistency, accounting for timing considerations, and allowing for iteration as necessary.

## Object Model Application

In the realm of computer science and software engineering, an object model is a conceptual framework that leverages object-oriented techniques to represent systems visually. Essentially, object models employ object-oriented programming principles to construct these representations. Examples of object models include DOM (Document Object Model) and COM (Component Object Model). When developing an object model, several key components must be considered, including interfaces, object references, actions, and exceptions.

In the context of the DriverPass scenario, applying an object model entail defining four fundamental elements for the model: abstraction, encapsulation, modularity, and hierarchy. This approach involves characterizing both physical and abstract objects within the system, while also considering the relationships between these objects as per the specific requirements of the DriverPass project.

## Process and Object Model Comparison

Process models and object models offer distinct advantages, and when considering the best fit for DriverPass, it's crucial to delve deeper into their respective merits.

Process models provide a visual representation of the system's layers, offering insights into the envisioned end product. By employing a process model, DriverPass gains a comprehensive understanding of its project's scope and identifies areas that may require revisions or patches. This clarity aids in aligning the project with business objectives. However, a notable disadvantage is the potential for over-analysis, which can consume valuable time and be costly, especially when time constraints are a concern, as evident in the Gantt chart.

On the other hand, object models offer visual representations of system objects, interfaces, actions, and exceptions. One significant advantage is their faster development process, aligning well with DriverPass's streamlined planning timeline. However, an object model may lack some structure and could pose a risk if the project involves inexperienced programmers unfamiliar with handling object models.

Considering the specific context of DriverPass, where time efficiency is essential, and the planning process is relatively concise, the object model appears to be the more suitable choice. Its rapid development aligns well with the project's timeline, making it a practical choice while still providing valuable visual insights. However, a combination of both models could offer a comprehensive perspective for the most effective project management.