# CS 255 Model Application Short Paper

Noah Archibald

noah.archibald@snhu.edu

Southern New Hampshire University

## Process Model Application

The most effective implementation of a process model into the design for the DriverPass scenario would be through the use of a flow diagram to model the fundamental interactions between the processes at work within the system. The flow diagram would start from the user end of their interactions with the system. The diagram could then be used to show the various branches to different users/functions within the system to perform various tasks and highlight key dependencies for each task along the way. For example, the user would be prompted to sign in/create an account and if they select sign-in, their credentials will then need to be authenticated and approved for a successful login attempt. The login depends on the authentication and vice versa and this relationship can be highlighted through the use of a flow diagram.

While a flow diagram is helpful in modeling the interactions between the various sub-systems at play within the DriverPass system, in its entirety, due to the complexity of the system and that it requires multiple steps for various processes, I would elect to create multiple flow diagrams for each major aspect of the DriverPass system. Having multiple flow diagrams that are much more concise and easier to comprehend, instead of one large and overwhelmingly complex model, would aid in keeping developers focused on each stage of development and ensures that the final product will be what was desired by the client.

Essentially, chaining tasks together via a flow diagram, while also roughly sketching out the logical flow and function of each part of the DriverPass system helps guide development of the system. Tasks can be broken up into achievable goals that can then be scheduled and worked into a given time frame and further allows for organizational tasks to be completed during the development process with relative ease. Overall implementing a flow diagram would be the most effective way to apply a process model to the development of the DriverPass system.

## Object Model Application

The best implementation of an object model into the design for the DriverPass scenario would be through creating a UML diagram. I could begin by defining the fundamental user class from which each specific class for each type of user would inherit its basic information. So, the fundamental user class has basic variables that contain the username, password hash, name, address, etc… and depending on the desired user type, a user class will have different capabilities and responsibilities based on their compositional relationship with the base user class. This compositional relationship also helps in modeling the processes that need to occur for each step in the chain of logic to continue within the system.

The student user would be comprised of various objects to hold data from the inherited variables from the base user class. For example, it could have a contact manifest object which stores all the basic user information, as well as a current grades object that stores the required variables needed to record and display a student’s progress in their current courses/tests. Also, since some roles, like that of the secretary or administrator, do not need to store personal information, their classes would not be composed of any objects but would still require functions that allow them to access the information objects stored in the user classes for maintenance and or modification purposes.

Since this only describes one aspect of the entire DriverPass system, it is easy to see how much more challenging it is to pull comprehension from simple text compared to a UML diagram. A UML diagram provides the necessary road map for full development of a system and in this instance, allows for the compartmentalization of various complex tasks that would be daunting to undertake without a concise representation of their various relationships with one another. A UML diagram in this instance not only serves to help describe the specifics of the DriverPass in system in detail but also allows for the fundamental aspects of the actual development of the system to begin as it names classes, objects, and variables and exhaustively chronicles the interactions between the various classes within the system and their intended purpose.

## Process and Object Model Comparison

An advantage of the process model, and a flow diagram more specifically, is that it very simply outlines the general flow of a program. A flow diagram can be used to effectively convey the development team’s interpretation of the client’s desires back to the client as an effective way of ensuring the project in development will be exactly what the client was expecting. Its simplicity is its greatest advantage, and it can communicate very clearly and concisely the functions of various parts of a system and their intended outcomes to someone will little to no technical understanding of the software development process. However, its simplicity can also be a disadvantage.

The simplicity of a process model can impart a general idea as to the flow of a system in its entirety, but it does not show how the system, or various aspects of that system, should be developed or designed in any technically rigorous way. There is no declaration of classes, objects or variables, or any explanation as to the hierarchical organization of the various processes within it. For a software engineer, a process model serves as a good starting point, but more information is required to build an effectively good representation of a system’s desired functionalities.

An advantage of the object model, or a UML diagram in this case, is that it more exactly represents a logical flow and relationship between various aspects within a system. For the DriverPass system this is vital, as it condenses the complexity of each individual function into a defined class with inherited objects and their specific variables. Representing the relationship between classes, be it compositional or associative, provides a depth of understanding that cannot be found when reviewing a basic process model for example. This is most valuable from a software engineers’ perspective, as it gives a more concrete foundation from which measurable goals can be drawn and decisions can be made during the initial stages of system development.

A disadvantage of an object model, however, is that it can be overbearing in terms of technical language, especially for a client with little to no software development experience and can lead to confusion and potentially strife between a development team and potential client. An object model also does not wholly describe the actions associated with each class or object and how their relationships would affect the eventual outcome of each step in the system. Most likely then, a combination of process and object models used together would help in cementing the understanding of how a systems developmental lifecycle will take place to a client.