# CS 255 Model Application Short Paper

Clifton Brann

clifton.brann@snhu.edu

Southern New Hampshire University

## Process Model Application

In the application of a process model for the DriverPass scenario, in this case I would start with a list of all activities the system is supposed to do, based on the data I have been given. In this case I would read over the available materials several times, and then start taking notes on what the system requirements are to design a system that will work for the chosen person. I would determine a list of functions or processes that the system needs to perform.

The next step would be to take the list and put it in a logical order and group related activities. This step would be iteratively performed in rounds or cycles to make sure all the activities were ordered and grouped accordingly. Then I would start drawing the process model chart beginning with the most general process steps: login/logout verification, end user functions, company employee functions, scheduling functions, banking/payment processing functions, database functions, admin/IT functions, security functions, and data backup functions. Each step or function would be arranged to interact with other units in a logical manner, with the connection representing the types of relationships between the different processes or steps. And for each of the broadly defined listed steps I would decompose each single general step into a well-defined process, again in an iterative process. I would look at each process in order to see what the process can be broken down to, and each round add another step at each general step, slowly making the whole system more and more complex. I would proceed in rounds as an iterative process to develop all the involved steps in an even and coherent manner, instead of falling down a rabbit hole with each function or step.

Then once the process model is fully decomposed, I would double-check to see that each necessary item is included from the list generated in the first step. As the individual general processes were decomposed, different steps or processes like: such as account creation, login/password verification, user side demographic data updating, user appointment creation, modification, and cancellation, payment processing, online test taking, data and access logging, employee/company side login, company side employee demographics, company side driver/trainer, car and student scheduling, employee accounts/files, etc. would be identified and simplified by smaller and smaller simpler steps. As necessary, I would break the very specific decomposed process model into parts, along the lines of related items or tasks. For example, the end user side data, demographics, appointment making, etc., would mostly be contained by one branch of the process model. That is not to say that there were no connections between the end user side and the rest of the system’s functions; there would be a significant amount of overlap for some modules. An example of this would be the payment processing module; anyone who needed to make payments would lose the same one, whether they are an end user or an employee, and since object-oriented programming is what we are dealing with, the whole system can use the same credit card payments module.

## Object Model Application

The application of an Object model to the DriverPass scenario would start by reading the interview transcript and taking notes on the different ideas, needs, and data that are discussed in the interview. In identifying what data is needed and what it will be used for, the designer can form objects that will represent the different processes of the system, and find what data should be in the object, and what the functions of the object will be. For instance, an object User will contain the attributes: first name, last name, login, and password. Then there would be objects that are children of User, like Client, Employee, Administrator, and Owner, which would contain all the attributes of User as well as further attributes; for example, Client would contain credit card information for payments, their demographic and contact information, a progress log of online courses taken and passed, and hours logged for on-the-road instruction, and a log of appointments that were made, changed, or cancelled. An Employee would have attributes like their demographic information, direct deposit information, employment records, and system access permissions.

Once all the objects were identified, and attributes were considered and assigned, then relationships would be charted by drawing connecting lines. The relationship between objects could be as simple as a line that shows that the objects are loosely connected and will share some data, or could indicate inheritance from a superclass to a subclass. The relationship lines can also show the multiplicity of the relationship, whether the relationship is a one to none, one to one, or a one to many situation. The explicit expression of the relationships between the different objects would help the designer to better visualize how the system will work, how data will flow through the system, and what will happen to the inputs at each step in the system. When creating the object model and drawing the relationship lines between the objects, an unused object can be found and eliminated if appropriate, or connected to whichever object they should be connected to. After the object model is complete, it should be clear what the relationships between the objects are, and how data should move through the system.

## Process and Object Model Comparison

Each of the different models has its strengths and weaknesses. The process model shows the path data should take through the system, and the object model shows the parts that process the data are related. I tend to liken the process model to a map; it lays out where you start and where you end up, and the steps or decisions it will take to get from point A to point B. The object model looks at the same system from the different perspective of how the different parts of the system interact with each other, and what is involved in making the system work.

As a tool to break down a system into its various processes and order of operations, the process model has the advantage over the object model because the process model will show how the system is supposed to work, whereas the object model will show what the parts of the system are supposed to do. The object model has the advantage of being able to express how the processes are supposed to function, what data they need, and how they interact with each other to make the system function. The object model shows a breakdown of the system in parts, with the processes as classes, the required data in the object in the form of attributes, and the data-changing process listed in the function section of the object. The disadvantage of the process model is that in creating the most detailed level may not be necessary, leading to wasted time in investing too much time in levels of detail that are not needed. It can be hard to know the proper level of detail to develop a process model to, for it to be maximally useful, without consuming too much time or resources. The disadvantage of an object model is that without a thorough knowledge of the desired product, it can be very difficult to be assured that all the appropriate objects have been included and properly connected and relationships expressed properly.

I view both types of model as tools in a toolbox and need to be applied when appropriate. Neither model is a perfect representation of the system to be built, but without a process model, it would be difficult to build an object model. It’s not that they are dependent on each other, but they are closely related, by being based not only on the same project and information, they also show different facets of the same system. Each model will offer its own insights, informed by the frame of reference. A designer should be wary of under- or overdeveloping each model. The model should be developed to a point where it is maximally helpful, without being so overdeveloped that it contains so much data that it is impossible to see the whole for the number and variety of the parts.

## References

Dennis, A. (2011). *Systems analysis and design with UML*. Wiley ; Chichester.

‌