# CS 255 Model Application Short Paper

Eric Boilard

Eric.boilard@snhu.edu

Southern New Hampshire University

## Process Model Application

I would apply use case modeling and activity diagrams as a part of my process modeling design for the DriverPass scenario. Use case modeling and activity diagrams really act as the framework from which we can build a successful system and going through the process of completing these items is relatively simple and useful. The use case diagrams are representing the way a business interacts with its outside environment. In other words, it is helping us understand what is going on from a actors perspective. An actor could be someone from management or it could be a user, but identifying how they interact with the system is the goal of this process model. Activity diagrams are a representation of the primary activities in a system and relationships they have with other activities that support business processes. Activity diagrams are very similar to use case models but focus more on the activities that are taking place in the system.

In applying these process models we will need to first make sure that we have determined our requirements and make sure we have any other information from the client. Then we will start creating our use case models and this task is relatively very simple and starts with first placing and drawing our use cases on the diagram. These use cases would include some of the major operations of the system so things like create appointment, change appointment, delete appointment, edit profile, select package, pay for package, and everything else that is an action. Then we would place our actors down and since we know the only different sets of people who are interacting are the customers, the head of IT and CEO, and then also the DMV since the DMV does play a role in relaying updated information to the system. After our use cases and actors are placed down we would then place down our subject boundary which helps us separate use cases from actors. This just helps divide up the model so it is easier to read. Then we have our fourth and last step which is to add some associations between the actors and the use cases to show the type of relationship we are looking at. This model is going to help us understand from a birds eye view the type of actions that are going on and who is doing these actions along with showing some easy to understand details in the form of the associations between actors and use cases.

Once our use case diagrams are in place and we have a general idea of what types of activities the actors are taking part in and in what scope. Now we can start creating our activity diagram which sounds like it would be very similar to our use case diagram, and that's because it is. The activity diagram is a diagram that shows the primary activities and the relationship among the activities. So, what's left out is the actors we labeled in the diagram before and instead we are focusing on the activity to activity relationship in the process. We first start this activity diagram construction with picking a process to diagram. This could be any from the previous diagram but we will use the creation, editing, or deletion of an appointment. Then we identify the activities that are needed to complete the process in questions. In either the customer or admin situation, there must be a log in, then actions like create appointment, delete appointment, or change appointment. If the log in was valid, then you would have access to those three actions. Then once those actions are done then the database must be updated and then the menu could return back to the main screen. Additionally, this activity diagram should be done for the entire login process and also for the action of choosing what package the customer wants to purchase. This should even be done for when the DMV has information they need to update. Then we identify the control flows which so the direction in which these activities are completed and the nodes to show the start and stop of the process. Then we include the object flows and nodes to show support for the logic. Lastly, but not least, we lay out the entire diagram to document the specific process. What this activity diagram accomplishes is that is breaks down the processes that we identified in the earlier diagram and then shows the relationships among the activities and actions that are taking place. This way we can understand the flow and logic behind each and every business process.

## Object Model Application

I would apply class diagrams as a part of my object modeling design for the DriverPass scenario. A class diagram does a great job at showing the viewer the static structure of an object orientated model. When we look at class diagrams they really have three main components that represent objects. The first is called the class or class name and this represents the objects themselves by type. For example we can make a class called customer for the DriverPass scenario and each object we create is an instance of this class. Within the class we have attributes which are the items that make up our objects. For example our customer class would probably have attributes like first name, last name, billing info, customer id, and status, just to name a few. Next we have our operations and these are actions that can be performed on our objects. These operations are usually constrained to updating, creating, deleting, and querying.

For our DriverPass project we would have multiple objects including Customer, Admin/IT, Order subscription, Instructor, User, Employee, and appointment. The customer class will contain attributes like email, password, phone number, billing info, and payment info. The admin/IT class will be accessed by the CEO and the head of IT and it needs to be able to see a lot of information. The Employee class won't need to be able to see as much as the admin/IT class but will still need to be able to share some basic operations. The Instructor class will need to be able to confirm appointments and see availability. Knowing this, the Customer, Employee, Instructor, and Admin/IT classes will all be subclasses of the superclass called user. This is because all four of these classes are considered users but play different roles. Some further investigating can be done to find out if the instructor class and the employee class can be made into just one to help with simplicity. Then we need to make an appointment class since the appointment is an object. This will contain attributes like instructor name, date, customer name, time, and confirmation. Since the whole DriverPass project is based on some sort of user making, altering, or deleting appointments and then the subscriptions involved then we know we need classes for all of these objects.

The operations will vary based on all of the classes but there will be some classes that share similar operations that can be reused. Every user is going to need a login and those logins need to be verified so that is one operation that they can all share. The Admin/IT, customer, and employee all need the ability to create, change, or delete appointments so those are operations that will be shared amongst those classes. Since the appointment class holds the appointment information than the only operation that it will seemingly need is one to update or change the information. Once we have all of our classes created with the proper attributes and the proper operations we can focus on creating the relationships among the classes. Since the customer, admin/IT, employee, and instructor classes are all forms of users we can simply apply a many to one relationship. For appointments, we should be able to apply a one to one relationship between appointments and customers which would suggest that only one user can occupy that appointment for that instructor. The appointment will also be tied to the instructor because there could be multiple appointments scheduled for a time if there are multiple instructors. This means our appointment class will have a relationship with the instructor class and the customer class. Since this relationship is going to be very important we could probably consider this a stronger relationship and mark it with a solid diamond on the customer and instructor ends. When this class diagram is finished and constructed it should provide a detailed overview of all of the object and classes in the system, what kinds of attributes make them up, what kinds of operations can be performed on them, and finally what types or relationships they have with each other. This will also serve as a blueprint for the actual development of the project as it is a detailed OOD concept that can be understood in OOP principles.

## Process and Object Model Comparison

There are some very clear and some not so clear disadvantages to both models for DriverPass. The process models offer a very simple view of the system. One process model helps us understand the relationship between actors and activities which helps us understand what every individual has as a role. The other process model shows us the relationships and movement of the activities being performed in each use case. These models both don't go into very much detail about the objects or the fine detail behind how the process will be programmed. Instead these process models offer an outside view that helps build an understanding for the early stages of the project itself. These process models are great for understanding the simple workings of the process and how the activities are being organized. They are bad for understanding the finer detail needed to actually build the system as they don't contain any information on data flow or objects.

The object models are a great tool that can be used as a blueprint for later development and are great for when designing needs to be reusable. That being said they are not so great for when we want to understand simpler relationships about the process. Our class diagram offers a great amount of detail of the classes, objects, attributes, and operations but this information is more useful in a programming understanding of things. If you were to show this type of diagram to someone like the client, then they could be very confused and not know what you are trying to say. For this reason, simpler diagrams, like use case diagrams and activity diagrams are used to help convey the process in simpler terms while bringing meaning to the process into complex enough terms to be able to start constructing the next set of models. Object models are considered a later step in the development process as will so by nature their contents will contain more detailed information that can be used for the development of the end product.

Together, these two sets of models are fantastic because they provide almost an entire overview of the system that is to be created. On one end we have diagrams that show us the relationship between activities and actors like customers and administration. On the other end we have diagrams that give us a detailed understanding of the objects and classes involved in the system so that we can proceed with creating it through code. Separately, these models can only accomplish so much as one does not include as much detail as the other and one is so full of detail that without being able to understand the diagram it might seem difficult to process. So, even though they are different in the information that they portray, they work together fantastically to help understand a system from beginning to end. Even though they are different, they do share some similarities. For example, our activity diagram and our class diagram are both diagrams of a process. The activity diagram is just a finer look at a specific process and the tasks involved in it and the class diagram takes a step back and focuses on multiple processes in the form of objects and operations. Now, the relationships will not be the same but they still follow the same principle of there is a path to follow and these are the ways you can get there. All in all, these two sets of models should work greatly in helping us create the DriverPass system and provide a great amount of detail scaled across all ranges.

## References

Dennis, A., Tegarden, D. P., & Wixom, B. H. (2012). *Systems analysis and design with Uml, 4th edition*. John Wiley & Sons.

Valacich J. S., & George J. F. (2019). *Modern Systems Analysis and Design*. [VitalSource Bookshelf]. Retrieved from <https://bookshelf.vitalsource.com/#/books/9780135172841/>