Adam Sissoko

November 26, 2021

Model Application Comparison

While the prompt suggests developing each model separately, and using one or the other during the development life cycle of the app “DriverPass”, I feel that it would greatly benefit the development team to spend resources to make both models. While this may require a bit of extra “front-loading” initially, it will undoubtedly save time on the backend of development. Overall, this will result in a faster development time (or at least a development time that is on par with what is expected), and thus the project will remain under budget. For the sake of argument, however, I will answer each prompt below, followed by my comparison, and finally, with my explanation as to why both models will be beneficial to the project.

Starting off with the process model, the first step would be to identify the process we wish to model; in this case, the DrivePass application’s user interactions and data flow. To do this we will revisit the gathered requirements from the initial meeting with the CEO and IT operative, as well as any use-case’s developed from these requirements. Once the process has been identified, we will then define the processes, the data stores, sources of data, and the flow of that data within the system. When each of these is determined, the model can be constructed in a visual representation. This Data Flow Diagram (DFD) will then be used to develop the individual processes that will run the system. Each process will be encapsulated in it’s own object, and data will flow in and out as specific data-types.

The object model for DriverPass will be slightly more detailed than the process model as it shows more detail about the objects and activities found within the process. For this, development of a DFD will be necessary (using the same one from the previous section will work). First, we will identify the objects of the system, and what attributes they will need to have. We will determine if these attributes will be visible outside the object itself, and what methods within the object will be able to manipulate the attributes. We will then identify the activities of the process, and what way they will manipulate the data that is passed to them. Additionally, we will determine what form the data will be in when it is passed to the activities. Once this object model is constructed, the development team can create each object per the requirements that are listed in the objects themselves. So long as the model is followed correctly, all objects should work together to complete the system once they are compiled with one another.

Both modeling approaches are beneficial for different reasons. Both models require the same initial resource gathering, but the process model is much simpler and therefore would be created more quickly than the object model. The lack of explicit details would make creating this model much faster, but ultimately would require developers to collaborate more with one another when they begin writing the code for the application. Collaboration is of course encouraged within the team, but more meetings about the application takes away from actual development time, or requires an increase in the time budget. On the other hand an object model approach requires a bit more initial front-loading, but time is saved on the backend. This happens because the object model explicitly tells the team how each object will behave, so the guess-work is taken out of that aspect of the development phase. Instead, developers can focus on other details such as how the functions will operate and manipulate the data.

Both models are important for developing software in their own respects, but put together they can potentially shorten the overall life cycle of the development of an application. An object model is very detailed, but creating one from scratch would require a lot of extra work initially to determine the data flow. Therefore, it must be assumed that even taking the object model approach would still require a process model approach initially to determine the overall flow of data. Devoting time to this phase will ensure proper data flow between predetermined objects and data sources / stores. Once the DFD is created with attention to the overall processes’ flow details, then the object model can be developed with input from the development team about how the objects should handle data transfers and manipulations. Taking the time to do both models will save time on the back end, as there will be little need for collaboration between team members regarding how the objects and data will interact with one another. This will ultimately save time during development, and ensure the project remains within budget.

In the case of DriverPass, it is recommended that the team first develop a process model for the system they are to develop. This will require input from all members of the team, including developers and testers. Once the model has been created, and approved, the object model can be developed by adding finer details to the process model. Again, this will be done with input from all team members, but especially the development team since they will be more familiar with methods that work best for handling the data. Once this model is complete, development can begin on the application.