CS 255 Model Application Short Paper

Matthew Muller

matt.muller@snhu.edu

Southern New Hampshire University

Process Model Application

Process modeling is an extremely valuable tool to help the development team visualize how an information system should function. Process models depict how data is inputted into a system, the processes within the system that transform it, and how it is outputted from the system. They are often graphical representations that show the way that data is distributed and transformed through the various interactions between the system and its environment, as well as between the different components of the system . One common example of a process model is a data flow diagram. A data flow diagram, or DFD, represents the data flows, data stores, processes, and external entities that interact with a system (Valacich & George, 2020, Chapter 7).

In order to apply a process model to a design for the DriverPass scenario, I would create a number of DFDs to help myself understand and visualize how the system will receive data, what functions the system will perform on that data, and how that data will be transformed into a system that has all the functionality that the company desires. In DFDs designed for the DriverPass system, the data sources would be the customers, the DMV, and the company's owner, IT officer, and secretary. To begin the flow of data from one of the sources into the system, the user must first go through the login process. In this process the user will provide their username and password and this data will flow to the verification process where the information will either be rejected in the event of an incorrect login or the system will grant the user the proper level of access for their credentials. Some of the other processes that a DFD for the DriverPass system would include are appointment scheduling/modification, account registration, addition/elimination/modification of users, business-customer contact, accessing of online classes/practice tests, enabling/disabling of driving packages, password resetting, updates to course content/questions, and system modifications/maintenance. Some data stores that would be shown in a DFD for the DriverPass system would be the databases required for customer information, course content/questions, drivers and cars, student grades/progress, appointment calendars, driving packages offered, and user information for employees.

In order to most effectively apply process modeling to the DriverPass system, I would have to construct multiple DFDs. There would be one that shows the flow of data for user login, one that shows the flow of data when users access course content/practice tests, one that shows the flow of data for the scheduling of appointments, one that shows the flow of data for updates received from either the DMV or IT officer, one that shows the flow of data for user registration, and one that shows the flow of data for the Owner's interactions with the system.

Object Model Application

Object modeling is very helpful when it comes to designing a system under an object-oriented approach. It helps to provide an easy to understand guide to the relationships that the classes and objects within a system have with each other and the impact that these relationships have on the system at large. This is extremely valuable especially when it comes to complex systems where a lot of interaction between classes is required. One common example of an object model is a use case diagram (Valacich & George, 2020, Chapter 8).

In order to apply an object modeling approach to the design of the DriverPass scenario, I would create a series of use case diagrams that will depict how the system will behave in order to complete the real-world tasks asked of it. This would include all of the different entities that act on the system, the actions that they perform on the system, and the way that these actions are related to each other. The different possible actors on the DriverPass system would be the customers, the DMV, and the company's owner, IT officer, and secretary. Some use cases that I would create would depict the login/verification action, accessing online courses and practice tests, scheduling/modifying appointments, registering, resetting a password, updating of the system, enabling/disabling driving packages, and business-customer contact. This will help the development team to understand how the system is supposed to deliver the functionality that the company and their customers require.

Process and Object Model Comparison

While the process model shows how the DriverPass system receives and outputs data, as well as the processes enacted on it in between, the object model helps to illuminate the details of how this is done. One of the main advantages of process modeling is that it can provide the development team with a clear view of the path that data takes in a system like DriverPass. They are able to see where specific points of data enter the system, how the system transforms this data, and what desired action of the system is achieved using this transformed data. One of the main advantages of object modeling is that it will show some of the inner workings of the system that make these processes possible. By helping the team to understand how different classes/objects within the system are related to each other and how they interact, the team can more effectively begin to translate the system’s desired processes into the code that will perform them.

One disadvantage of applying a process model to the DriverPass system is that it does not provide much insight into the difficult task of automatically receiving updates from local DMVs. For this, the team will need to know how the data systems maintained by the DMV will interact with the DriverPass system. Also, since a number of separate models will be required to show the different processes, pursuing a process modeling approach may be time-consuming and complex. A disadvantage of applying an object modeling approach to the DriverPass system is that it can be difficult to account for unpredicted interactions between the system and its actors. If the use cases are not thorough in depicting the functionality of the system, it could lead to problems in the system when an unexpected action takes place.

References

Joseph S. Valacich, Joey F. George. (2022). *Modern Systems Analysis and Design*. Chapter 7: Structuring System Process Requirements. (pp. 203-215). Pearson.

Joseph S. Valacich, Joey F. George. (2022). *Modern Systems Analysis and Design*. Chapter 8: Structuring System Data Requirements. (pp. 266-280). Pearson.