Name Of Use Case Models Was Built Upon Refund And Cancellation of An Order

Model 1: Conceptual Class Model

The first model is the conceptual model of our refunding system. This model shows the relationships between the classes in the system and the functions each class has For this model, we start at the order class which stores all the information for the order and has an aggregation relationship with the product class as an order can be made up of many products. Additionally, the order class has a direct relationship with the payment information because payment information is stored when an order is made. To build upon, the order then moves to the request class which it shares a direct relationship with, the request class stores 2 constant int values which correspond to the allocated time for making a refund or cancellation. The next 2 classes are identical in the sense that they both have a dependency on the request class. The refund class and cancel class both contain functions that determine if the request pertaining to them is valid. Lastly, the payment class is dependent on the payment information class in order to get the payment information from the user so the money can be returned back to the user.

Observations:

The model was updated as well with the use case when we took into consideration that there may be an issue with returning the money to the user. So, if the money cannot be returned to the user we will give them in-store credit as a replacement. The function for sending them the in-store credits can be found in the payment class.

Model 2: System-Level Diagrams:

This diagram shows the process in which components of the website communicate with each other after a request has been made.

There are 2 pathways the system-level sequence diagram can follow. The first is if the request is valid the diagram will show how the information reaches all the way to the bank and how the message is received.

The second shows how the components of the website communicate with each other to reject this request.

Observations:

An issue we found with this model is that it wasn't updated after the use case scenario was updated, so it does not show the relationship of how the user will receive in-store credit. This model assumes that there will be no issue if the refund request is valid.

Model 3: State-Level Diagrams:

This diagram will show the behavioral changes of the system when a user makes an order and/or refunds said order.

This diagram has 3 exit points.

The first occurs when the order is received with no issue.

However, if there is an issue then the behavioral state will change, and depending on the request it will go to the second or third exit point. If the request is invalid it will reach the second exit point titled “request rejected” and the pathway ends. However, if the request is valid it will reach the final endpoint titled “refund issued”.

Observations:

This model is the only one that incorporates the warehouse. As stated in the use case the refund will be issued when the product is received in the warehouse. The class model and sequence diagrams do not display this relationship but the State level diagram does.

Conclusion:

Within the three models discussed for the refund/cancellation of an order, they depict the relationships between the classes, behaviors, and interactions between the components of the system. Although, all models do not depict every single aspect of the system, when they are used together they explain how the system will work according to the Use Case they were modeled after.