**Evaluate an Object Model**

Justin Starr

Department of STEM, Southern New Hampshire University

CS 255 – System Analysis and Design

Professor Kim-Marie Foss

March 26, 2023

# Evaluate an Object Model

* What are the different functions of the online storefront? How are they represented in this type of model?

Each of the seven objects represented in the Online Shopping Cart Object Diagram has functions listed in the bottom third of each class. Each class is representative of its object. The functions for each of the classes are as follows:

* Customer Class
  + public register()
  + public login()
  + public updateProfile()
* User Class
  + public verifyLogin()
* Administrator Class
  + public updateCatalog()
* Shopping Cart Class
  + public addCartItem()
  + public updateQuantity()
  + public viewCartDetails()
  + public checkout()
* Order Class
  + public placeOrder()
* Shipping Info Class
* public updateShippingInfo()
* Order Details Class
  + public calcPrice()

Functions can be represented as either private or public. A minus sign in front of the object means that the function is private and only accessible by that object/class. Public functions, which can be accessed from outside of the object it belongs to, are represented with a plus sign in front of the function title.

* What are the different classes of “users” represented by this object model? What are the associations between these classes?

In the Online Shopping Cart Object Diagram, there are two different types of users, Customer and Administrator, both of which inherit from its superclass, user. In the Online Shopping Cart Diagram, inheritance can also be seen or depicted by a hollow arrow that starts from a child class and ends at its parent class; the parent class ending with the arrow facing it. When a user logs in, the method verifyLogin() would be called, and a Boolean value of true would be returned if the user is verified, or a false value is returned if the user is not verified. Depending on the type of user that is verified upon user login, a customer would be able to access customer attributes and methods, and vice versa if an administrator logs in, that user would be able to access administrator attributes and methods. Both types of users are able to access the User class/object attributes and methods because of their inheritance.

* How would the objects “use” their respective variables and functions?

Similarly, as described above, for how the Customer and Administrator classes/object are able to use the methods and attributes of the User class, Shopping Cart and Order inherit from Customer, allowing both methods to use the Customer attributes and methods along with their own. Shipping Info and Order details inherit from Order, allowing these two classes to use attributes and methods not only from their parent class, Order but also from Order’s parent class, Customer. Important to note also is that on the diagram, we can see that both Shopping Cart and Order classes indicate the use of a none-to-many model. The Customer, Shipping Info, and Order Details class indicated by the number one that there can only be one instance of each of these classes/objects at any time.

* Does this object model capture all of Hamp Crafts’ desired functionality? Why or why not?

The first thing that I notice that the object diagram does not specify, depending on how a person might interpret the object diagram, is that it does not include the use of debit cards, which is specifically mentioned by the client. If I were writing the diagram, I would make sure to include the word “debit” in the private attribute creditCardInfo. For example, I would say creditDebitCardInfo. Another aspect of the diagram that I do not believe captures all of the requirements of the client is how they would like to utilize an established credit card vendor, such as Square or Shopify, for example, to receive customer payments. The shopping cart does have a checkout method; however, I would want to likely have a checkout class that inherits from the Shopping Cart class. Then you could specify in the diagram the various ways the customer is able to check out by listing new attributes and methods in the checkout class. Also, the client specifies that the customer should receive a notification based on the information in their personal profile regarding order status and confirmation. The Order class does include a status attribute; however, I believe it should also have a method for sending the desired notification to the customer. Also, if I did implement a checkout class, I could have a similar method here for when an order is placed; it also sends a confirmation notification to the customer. Lastly, I do not think the Administrator class fully captures the desired functionality for administrative users. The updateCatalog method is the only method listed for administrators., which only allows support for the website itself. I would also include additional methods that would provide specific support for customers as well.

* The above diagram uses a solid diamond shape to represent a form of aggregation. What type of aggregation does this represent? What does it imply about the relationship between the classes? Why is a solid diamond the appropriate choice here?

Aggregation is a specific type of association between two classes/objects that specify a whole and its parts (Lucid Software, 2017). The type of aggregation we see being represented in the Online Shopping Cart diagram is called composition. In this type of aggregation, it means that its part cannot exist without its parent class (Lucid Software, 2017). In our diagram, we see this in four different instances. Shopping Cart cannot exist without a Customer object. An Order object cannot exist without a Customer object. Shipping Info cannot exist without an Order object. And lastly, Order Details cannot exist without an Order object. The reason why a solid diamond is appropriate instead of an unfilled diamond is that the unfilled diamond represents a type of relationship or aggregation where either can exist without each other. For instance, if the diamond from Shopping Cart to Customer were not a solid diamond, the diagram would then be indicating that a shopping cart could exist without a customer. However, we know that this could not be true for the client’s desired system. Without a customer, there would be no need for a shopping cart. The same applies to all objects in the diagram that utilize composition.

* How well do you think a process model describes the system? What information does it make easier to understand? What aspects of the system are more difficult to understand or are not represented?

I think as it concerns describing a system, I think a process model does so in its own unique way versus an object model. The process model is beneficial because it describe the structure of a system, what information is used by the system, and how that information is handled. The information that it makes easier to understand is not the specifics of the information that the system utilizes but the kinds of information that the processes will handle. By types of information, I mean that if a process diagram has a process Check Out and a sink/source carrier and shipper, we will be able to see that Shipping plan information flows from the check-out process to a sink/source carrier and so on. On a process model, I would not see the specifics of the Shipping plan information; I would simply know that this particular piece or type of information is flowing from the process to a source. Because only this type of information is given on a process model, someone reading only a process model may not fully understand the system because the specifics of the information are hidden. It almost leaves out details, for example, such as how information needs to be protected.

* How well do you think an object model describes the system? What information does it make easier to understand? What aspects of the system are more difficult to understand or are not represented?

Contrary to a process model, I also think an object model has a unique way of describing a system, but without both a process and object model, the system as a whole would be difficult to understand fully. The object model doesn’t capture the full flow of data. It simply illustrates how objects in the system interact with each other. This is important, though, because we can better understand what parts of the system can access other parts of the system and its information (attributes and objects). We also get a better understanding of how data can be protected through inheritance as well as through encapsulation by classifying objects and attributes as private, public, or even protected. I think it makes it easier to understand what objects are able to access this information, thus enabling us to have another view of processes because we are able to see where data flows can or cannot occur within processes.

**References**

(Lucid Software, 2017, July 21). UML Class Diagram Tutorial. Youtube.

<https://www.youtube.com/watch?v=UI6lqHOVHic>