Hamp Crafts Online Storefront Object Model Interpretation

By: Darrell Walker

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

The online storefront offers various functions for both customers and administrators. Key functions include:  
- Customer registration and profile management: Customers can create an account, update their profile, and store contact, billing, and shipping information.  
- Shopping cart management: Customers can add items to their cart, adjust the quantity of items, view the cart, and proceed to checkout.  
- Order placement and tracking: Customers can place an order, track its status, and view order history through their profile.  
- Payment processing: Though the UML model doesn't explicitly outline this, integration with a third-party credit card service for payments is implied.  
- Administrative functions: Admins can update product catalogs and manage customer support.  
  
These functions are represented as methods within various classes like Customer, Shopping Cart, Order, and Administrator in the UML diagram.

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

- Customer: Manages customer profiles, shipping details, and payment information.  
- User: Acts as a general class for Customer and Administrator. It handles login verification.  
- Administrator: Manages the catalog and ensures the system runs smoothly.  
  
The associations between these classes suggest that customers interact with the storefront for shopping, while administrators manage backend functions like catalog updates.

# 3. How would the objects “use” their respective variables and functions?

Each class contains both variables and functions specific to its role:  
- Customer: Stores personal information like customerName, address, and creditCardInfo. Functions like register() and updateProfile() enable customers to manage their profiles.  
- Shopping Cart: Tracks items added to the cart and allows actions like addCartItem() and checkOut().  
- Order: Stores order-specific information and lets users placeOrder() or track it.  
  
These objects use their variables to store essential data and their methods to manipulate that data as part of interactions between users and the system.

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

Yes, the object model does a good job of capturing the essential functionality Hamp Crafts wants:  
- Customer account management is well-covered through the Customer and User classes.  
- Order processing is handled by the Order class and its associations with ShippingInfo and OrderDetails.  
- Payment processing is implied, though integration with third-party services is not explicitly modeled.  
- Administrative support is covered through the Administrator class.  
  
The only part that might need more explicit representation is the integration with third-party payment vendors.

# 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?

The solid diamond used in the UML diagram represents composition aggregation, meaning that the contained objects are a part of the parent object and cannot exist independently. This is an appropriate representation, as an OrderDetail is inherently tied to a specific Order.

# 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?

## Process Model

A process model focuses on the flow of actions or steps required to complete a task. In Hamp Crafts' case, a process model would make the following easier to understand:  
- Workflow of placing an order: This includes steps like adding items to the cart, proceeding to checkout, making a payment, and tracking the order.  
- Interactions between system components: How customers, admins, and the payment system interact over time.  
  
However, it might be harder to represent the static structure of the system, like the relationships between data objects.

## Object Model

An object model, like the UML diagram for Hamp Crafts, excels at showing:  
- System architecture and relationships: It clearly illustrates how different classes are connected.  
- Attributes and methods: It makes it easy to understand what data is being stored and how it can be manipulated.  
  
However, it can be less effective in showing how processes flow, such as the step-by-step path of a customer checking out or the interactions with external systems like payment vendors.