CARLETON UNIVERSITY
Department of Systems and Computer Engineering
SYSC 5708 (ELG 6178) Model-Driven Development of
Real-Time and Distributed Software
Fall'2014

ASSIGNMENT 1

DUE DATE: ~~Wednesday, Oct. 15,~~ Monday, Oct. 20, 2014 (Bring a hardcopy to class or e-mail a single PDF document including all the text and UML diagrams).
POSTED: Wednesday, Oct. 1, 2014.

Problem statement

A supermarket has a number of Point of Sale Terminals (POST). Each is operated by a sales assistant (cashier) and has a keyboard, a barcode reader, a display, a receipt printer and a scale for weighing products. The products are identified either by a general bar code attached to the product, or by a shorter store code entered at the keyboard by the cashier for products that don't have a bar code attached (e.g., fresh fruits and vegetables). Products are sold either by piece or by weight. The keyboard can be used to input details such as the number of items of each type being purchased (so that each individual bar code does not need to be scanned multiple times). The receipt printer is used to print a short name of the product, its item price, number of items (or weight), as well as the total price of the sale. The same information can be shown on the display sequentially (i.e., product-by-product).

All POSTs in a store are connected to a store server, which stores product information (such as description, price and quantity in stock). No product information is memorized at POST. The store server, in turn, gets the prices from a central depot server, from which it also orders goods to replenish the stock when necessary. A POST can be used by the store manager to monitor the total sales made in the store, the total sales made by each point of sale terminal and the current stock. (There is no other interface unit to the store server than the point of sale terminals.)

For the sake of simplicity, the ordering of goods by the store server has not been addressed fully here. You may assume that the store holds in stock all item types that it sells, and will reorder a suitable quantity when the stock falls below a certain threshold. However, no other details have been addressed.

The standard procedure of using a point of sale terminal when a customer wants to check out is as follows:

- The cashier records for each item its bar code, or its shorter store code, as well as the weight, or quantity (if it is greater than one).
- The POST displays the price of each item and its description.
- When all the purchases are recorded, the cashier indicates the end of the sale.
- The POST displays the total cost and starts the payment transaction.
- The customer selects his or her payment method:
    - Cash: the cashier takes the money from the customer and puts it in the cash tray; the terminal indicates how much change the customer is to be given;
    - Debit or Credit card: a banking terminal is attached to the POST. It sends a request for authorization to an authorization centre, according to the card type. The customer either enters the required information for the debit card, or signs the receipt for the credit card at the end of the transaction.

- The POST records the sale and prints a receipt.
- The cashier gives the receipt to the customer (who signs the merchant's copy if the payment was made with a credit card).
- When the payment transaction is finished, the POST sends the information on the number of items sold to the Store server.

Perform the following steps of the COMET methodology:

1. Requirements Model

a) Develop the use case diagram for the software system controlling a POST. Consider different alternatives and employ use case relationships as necessary.
b) Select two use cases that are more complex and describe them in English by using the COMET template.

2. Analysis Model

2.1. Develop the *static model*:

a. Identify the physical objects (classes) in the problem domain and build the corresponding class diagram
b. Develop the system context model
c. Identify the entity classes
d. Add boundary objects
e. Start developing the system structure, by grouping classes into subsystems. (Please note that the subsystem design will be finalized during the Design Model phase).

2.2. Develop the *dynamic model* for the system. For each of the use cases described in detail in step 1.b do the following:

a. Identify the participant objects (classes).
b. Develop sequence diagrams for the two use cases described in detail in step 1.b, describing the "happy path" and possible alternatives. (You may use *opt* or *alt* fragments for modeling the alternatives on the same sequence diagram as the happy path).
c. Identify information passed between objects, which may lead (or not) to the discovery of other objects.
d. Develop a statechart for the state-dependent controller involved in the realization of the selected use cases. Explain how the consistency between the sequence diagrams and the statechart is maintained.

Note that the statechart you'll build has contributions from the two use cases. However, the statechart will be incomplete, as other system use cases not detailed in the assignment may contribute to it.

What to hand in: For every step described above, hand in the corresponding UML 2 diagram and/or English text.

Guidelines

In software engineering, it is important not only to solve the problem, but also to convey your solution to other people. Therefore, try to make your assignment solution clear and neat. It is advised to use a word processor (as hand writing may be harder to read) and a UML editor for the diagrams.