Software Design (F28SD2):

EasyShopper – An Exercise in Systems Design

Andrew Ireland & Brian Palmer
School of Mathematical & Computer Sciences
Heriot-Watt University

January 16, 2012

1 Introduction

Imagine a shopping experience where check-out queues were a thing of the past! You simply fill your basket/trolley and proceed through a check-out area where your shopping is automatically priced. The technology to realize this dream exists today, a key ingredient is the Radio Frequency Identification (RFID) tag – a smart product label that encodes product, price and security information. Your task is to design a RFID based shopping system which will be called EasyShopper. The requirements for EasyShopper are detailed in §2, while in §3 and §4 the tasks and deliverables that are expected of you are described respectively.

2 System Requirements

The EasyShopper system will involve a number of distributed components, e.g. tags (RFID), sensors, databases, transmitters, receivers, controllers and actuators. As noted above, the tags are the key ingredient of the system – each tag has a:

- 16 digit identifier (unique to tag).
- product code (unique to product).
- product status – *new* or *returned*.
- price data, which has two parts – *value* and *currency*.
- security status – which can be either *active* or *deactivated*.
- error status – which can be either *good* or *defective*.

When products are delivered to a shop they already contain a tag. However, the tags need to be updated with the current price data, for the product, and be activated. This process takes place just before the products are brought onto the shop floor for stacking, and involves the Tag Update Subsystem (TUS). The TUS transmits a "wake-up" signal to the tags that are placed in front of it. In response, each tag transmits its unique identifier and product code to the TUS. Once it knows the product code, the TUS requests the current price of the product from a central retail database. The tag is then updated with the current price, and its security status is set to active.

When a customer completes their shopping, they proceed to a check-out area where they pass the Product Check-out Subsystem (PCS). Like the TUS, the PCS interrogates the tagged products – it gathers data on the products and prices and generates a customer bill, including a list of all the products and prices. A standard chip-and-pin payment point is used to complete the transaction. Once payment has been confirmed, a receipt is printed and the tags security status is set to deactivated. If a tag is detected as being defective, then a shop supervisor needs to intervene and via the PCS, select the product code and

ensure the security status is deactivated. Note that the PCS is swipe-card protected to prevent customers gaining access to the system.

On leaving the shop, customers will pass a Product Security System (PSS), which detects any tags which have active security status. If such an event occurs then an alarm is sounded. If required, a shop supervisor has the authority (swipe-card protected) to interactively interrogate a tagged product and where appropriate override the security status, i.e. activate or deactivated.

Finally, a Product Return Subsystem (PRS) is also required. Located at the customer service area, the PRS will allow a shop supervisor to check the value of a returned product, and via a chip-and-pin device, refund the customer. Note that the PRS must also allow the shop supervisor to register the returned product as a "return" via the product tag.

3 Tasks

For the system described above, you are required to develop both *function-oriented* and *object-oriented* models of EasyShopper where appropriate. Specifically you are expected to complete the following tasks:

T1: Function-oriented perspective:

- **T1.1** Develop **data flow diagram(s)** where appropriate.
- **T1.2** Develop **structure chart(s)** where appropriate.

T2: Object-oriented perspective:

- **T2.1:** Develop UML **use cases**, both graphical and textual forms.
- **T2.2:** Develop a **class-responsibility-collaborator** model.
- **T2.3:** Model the static aspects of the EasyShopper system using UML **class diagrams**.
- **T2.4:** Model the dynamic aspects of the EasyShopper system using UML **state machines**, **activity diagrams** and **sequence diagrams**.

4 Deliverables

Your submission **should** take the form of a report (hard-copy) and **must** include the following:

D1: A statement explaining any assumptions you have made about the requirements.

D2: Your design models, using a structure that reflects tasks **T1** and **T2**.

D3: A short statement (no more than 100 words) on the strengths and limitations of the design notations that you have employed. Note that we are looking for a statement that focuses on the suitability of the notations highlighted in **T1** and **T2** for the given design exercise – and **not** the strengths and limitations of your actual design.

This assignment counts for 30% of the overall mark for the course and should be submitted via the course work box located outside the Student Office (room 1.24) by **3pm on Monday 27 February, 2012. Note that this is an individual project which means that your submission MUST be your own work.**