1 System Overview

The Blind-Accessible Shopping System aims to make shopping simpler for the visually impaired population by giving these shoppers directions to the products on their shopping lists. Additionally, the system will help ensure that the right products are indeed purchased by providing the necessary functionality to scan and identify the products that the shoppers pick up at the shelves. Drawing on the auditory facility that is often developed through many years of training for the blind population, the system will make use of Text-To-Speech software as its primary output vehicle. To complement this method of output, the system will allow users to input its simple commands by voice.

2 Hardware Architecture

2.1 RFID Subsystem

2.2 Barcode Subsystem

3 Software Architecture

Several architecture concerns arise when considering a system that depends heavily on inputs from hardware devices as well as making frequent use of a sizeable database. Regarding the hardware, a critical decision must be made about whether the devices will provide input when it is received via callback functions or whether the devices will instead be polled at regular intervals. In this particular system, the design team has agreed that the callback model is more appropriate for both the RFID and barcode reading devices. This decision is particularly important in regard to the RFID reader, as a polling architecture may cause the system to miss important positional updates.

In addition to the interaction with hardware devices, the architecture needs to accommodate simple methods of database access. As such, the design team was faced with choosing a development framework that would facilitate the development of a data and event driven application. The chosen framework is
the .NET Framework by Microsoft, within which the development language is C#. The .NET Framework was designed around the concepts of events and data access, making it a sensible choice. Additionally, Microsoft has developed the Speech API, which provides Text-To-Speech and voice recognition functionality and integrates seamlessly into .NET applications.

As mentioned previously, the Blind-Accessible Shopping System is designed with events and data access in mind. From a high-level viewpoint, the system will fire events when receiving input from the hardware devices, and these events will then be handled by a central event switchboard. Because the events are being raised by the input controller classes and subsequently handled by a different class, the whole system is somewhat of an exercise on the Chain of Responsibility software design pattern. Besides the event-raising input controllers and the switchboard, the system will also contain numerous classes for business objects such as products, store locations, and shopping lists as well as objects that generate and represent walking paths through the store.

3.1 Input

3.1.1 Hardware Adapter Classes

The system will use classes to handle the raising of events from the RFID and barcode readers. LocationReader provides the RFID functionality, and its key method, OnLocationRead(), fires an event signaling that a location (RFID tag) has been read. The LocationReadEventArgs class encapsulates the location that has been read to allow the event to carry with it some useful information.

Similarly, ProductReader contains the event functionality to indicate that the user has scanned an object with the barcode reader. The ProductReadEventArgs class encapsulates the UPC (Universal Product Code) of the product that has been scanned. The system will use classes to handle the raising of events from the RFID and barcode readers. LocationReader provides the RFID functionality, and its key method, OnLocationRead(), fires an event signaling that a location (RFID tag) has been read. The LocationReadEventArgs class encapsulates the location that has been read to allow the event to carry with it some useful information.

Similarly, ProductReader contains the event functionality to indicate that the user has scanned an object with the barcode reader. The ProductReadEventArgs class encapsulates the UPC (Universal Product Code) of the product that has been scanned.

3.1.2 Voice Recognition

The system raises events on voice commands through the SpeechRecognizer class. A generic voice command event, OnVoiceCommandReceived(), notifies the Conductor that the user has provided input. Like the hardware adapter classes, SpeechRecognizer utilizes a custom event argument
This class, VoiceCommandReceivedEventArgs, contains a value from an enumeration of the available voice commands:

- 'Directions'--User requests directions
- 'Location'--User requests current location
- 'Next'--User decides to skip searching for current item and move on to the next one

3.2 Output

3.2.1 Abstract Responder Model

The architecture includes an abstract Responder class. A simple application of the Strategy software design pattern, this class allows different output providers to be coded and used without altering the operation of other classes. This would be useful in the event that the user preferred output via a tactile Braille device or something of the sort.

3.2.2 Text-To-Speech Output

The SpeechResponder class controls the Text-To-Speech functionality of the system. Using objects and methods from the Microsoft Speech API, SpeechResponder controls reading output to the user.

3.2.3 Graphical User Interface

Although BASS is designed for use by the visually impaired, it includes a GUI component. The primary purpose of the GUI is to further testing efforts; however, it will also help store personnel assist BASS users while shopping. The Displayer class is the main Windows form for the application. Important information, such as destination product, current location, and last issued voice command will be displayed. As has been done in other areas of the project, the GUI will be updated with an event handling scheme.

3.3 Shopping List

3.3.1 Programmatic Structure

Most shoppers, visually impaired or otherwise, go to the grocery store with a list. In the case of BASS, shopping lists are invaluable. The structure BASS uses to handle shopping lists is a bit of an exercise on encapsulation, as the objects are very self-contained.

At the core of the shopping list is the Product business object, which has a similar structure to the Product table schema in the BASS database. This business object also contains some relevant information that is located in other database tables, in efforts to reduce the
number of database operations required to gather information for output. The companion to products on a shopping list is, naturally, the desired quantity for each product. Because the product has widespread applicability throughout the system, it would not be practical for Product objects to know the needed quantity. Thus, the ProductListing class serves as an adapter between the ShoppingList and Product objects.

3.3.2 File Format

For the purposes of testing, the shopping list will be constructed from a simple text file containing a product ID followed by a comma and then the desired quantity of that product.

3.4 Store Walking Path

3.4.1 Representation

Before representing entire walking paths, it is beneficial to create a simple, clean representation for the points in a path. Thus, the StorePosition class encapsulates the X and Y coordinates of a point in the store. Note that the coordinate system is a grid of arbitrary square size. This class also contains an enumerated value of type StorePositionType, which will be used to differentiate places in the store as planogram corner points, RFID tags, product locations, or store boundaries such as walls.

The Path class, then, uses the C# concept of generic programming to store path points as a list of StorePosition objects. Very similar in structure and operation to the shopping list, the path constructs itself with only the store's specified start position, which is retrieved from the StoreData object.

3.4.2 Generation Algorithm

3.5 Event Switchboard

3.5.1 Responsibilities

Because of the event-driven nature of BASS, much of the system's functionality depends on the event switchboard found in the Conductor class. This class is responsible for handling the events raised by other classes in the system. Additionally, it is the central repository for key runtime information, maintaining the user's shopping list, walking path, and current location in the store. Both input and output are event-driven, so this class plays an important role in these subsystems. Before any of this can occur, however, the Conductor must also initialize the ProductReader, LocationReader, and SpeechRecognizer objects so that the events will be fired.
3.5.2 Event Firing Scenarios