Homework 4

Assistant for Recycling and Waste Management to Reduce Recyclables in Landfills

Noah Beach, Kevin Christensen, Emmet Cooke, Jacob Leno, Caleb Scott
CS361 Group 18 - Spring 2018

1. Contributions Summary

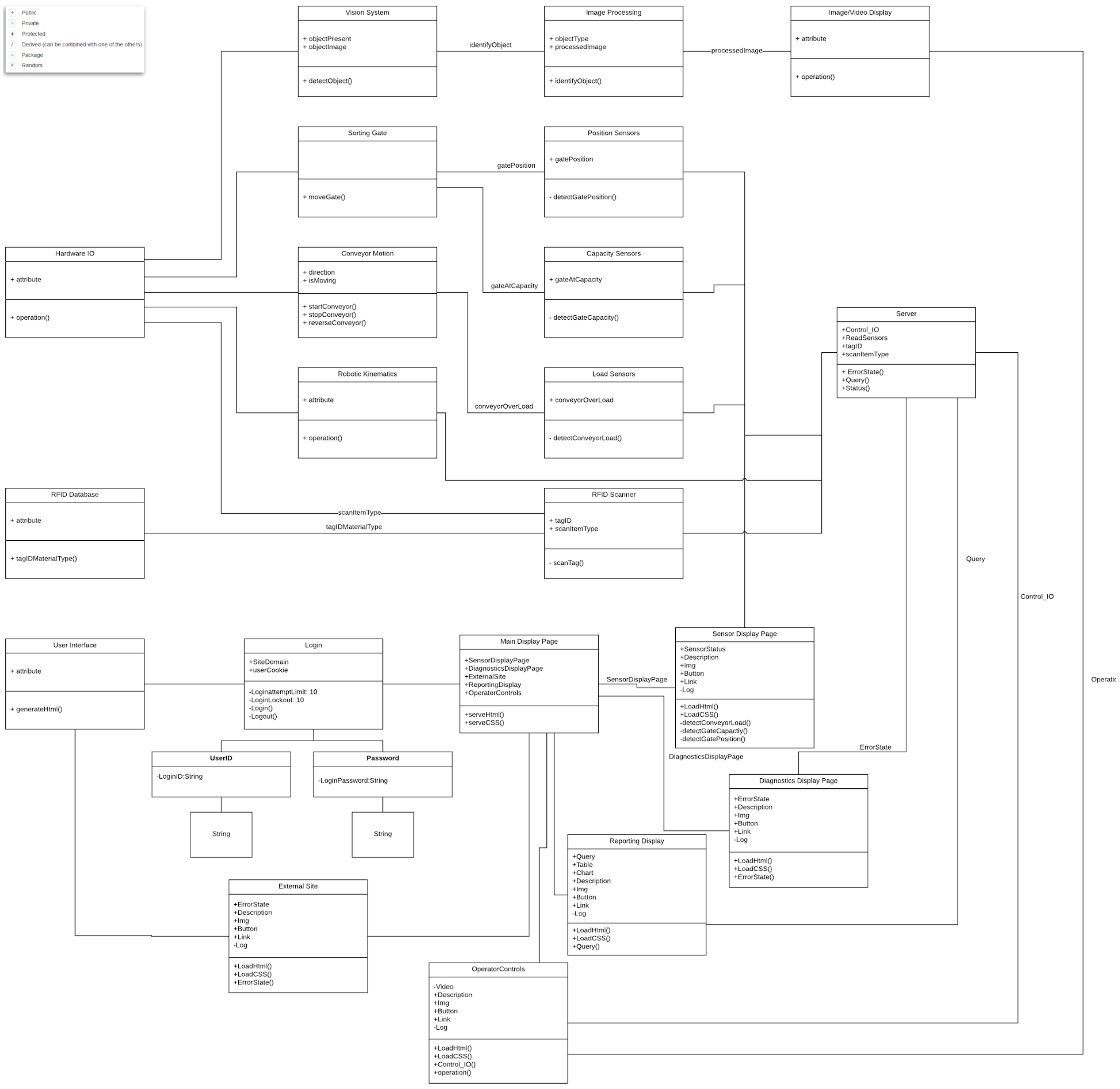
The work was divided among our group for HW4 as follows: Noah Beach - Package Implementation/Development, Interface Contracts, Kevin Christensen - UML Class Diagrams and Design Pattern Analysis, Emmet Cooke - Sequence Diagram, Jacob Leno - Package Implementation/Development, Potential Exceptions, Caleb Scott - UML Class Diagrams, Interface Contracts.

2. UML Class Diagrams

2.1 UML Class Diagram

The following image (figure 2.1a) is a UML class diagram for the Recycling Assistant system. It identifies the key object oriented entities, and their attributes, for the Recycling Assistant system.

Figure 2.1a - UML Class Diagram:



Note: A digital, high resolution version of figure 2.1a can be accessed via the following URL. https://drive.google.com/a/oregonstate.edu/file/d/11AHLgxccGOkAeN7PVlthuFPsrtxW1g9i

3. Package Implementation and Development Process

3.1 Packaging Implementation

The system will be implemented using Polymorphism where appropriate, allowing maximum code reuse during the design lifecycle. Examples of objects or modules that would benefit from Polymorphic design would be sensors sharing the same Polymorphic lineage, user interface modules would share the same Polymorphic lineage, as well as modules involved in mechanical sorting would all share the same Polymorphic lineage.

Based on the UML class diagram, there will be several units of the program that demonstrate either a type of cohesion or type of coupling. Some examples of this include the data coupling between the RFID Scanner and the RFID Database, where unstructured data (the value from the scanned RFID tag) is sent from the RFID Scanner to the RFID Database. Another example of data coupling would be the RFID Scanner and the Server, in this instance unstructured data (the value from the scanned RFID tag) is sent from the RFID Scanner to the Server which then makes sorting decisions based off that data. A third example of data coupling would be between the User Interface and the Server, where various bits of unstructured data (instructions, sensor info, etc) is sent between the units. With respect to cohesion, some examples of communicational and procedural cohesion can be found between the Conveyor Motion and Sorting gates, as well as between the various parts of the imaging system (Vision System/Image Processing/Image Video Display) as data is passed from one unit to another. In addition to those systems, the RFID Scanner has an indirection logical cohesion with the sorting mechanisms since the data it scans is eventually interpreted and used as a command decision for sorting.

In general the packaging schema shown via the UML class diagram promotes unit cohesion by grouping together the parts of the system that will utilize similar or the same data. By doing this the program's ability to be tested, maintained, as well as the program's general design is enhanced.

3.2 Development Process

Our design would work best with incremental development best because much of our code can be reused from one package to the next. Development of the code would also be easier with the incremental method because the most important parts of the system could be designed first and tested. The system is primarily build around the RFID Database and this database would have to work before testing could be done on things like sensors and the sorting gate.

The Hardware IO portion of the code is also a major part. The Hardware IO portion will have to be implemented at the same time as the RFID Database before other testing can occur. Once both the Hardware IO package and the RFID Database have been built we will be able to work in tandem with mechanical engineers to create software for smaller portions of the project such as the actual RFID sensor. Therefore, it would make sense to build the large portions of code necessary to the project and add the smaller less critical portions of the project on through incremental means.

An incremental process will be well adapted to code reuse in the project. Packages for sensors, user interface, and mechanical sorting will be polymorphic and will be reused for the project. An incremental process will allow us to build upon old code and add to it as we need. This will be useful in areas such as the vision system and RFID and position sensors. The Vision System could work by using a base class that initially prepares the image and detects if an object is present in the video stream. The Image Processing package would be built off the Vision System package and use polymorphic functions to further identify the object. Finally the Image/Video Display package could then be a further refined version of the original Vision System package used to make a decision on what to do with the object and display the image.

4. Design Pattern Analysis

4.1 Design Pattern Analysis

One design pattern that could prove particularly useful in this system is the template design pattern. With many of the different sensors, there is a detectX function that is looking for some sensor output state. By implementing a template pattern for sensor detection, the maintainability of the system could be improved.

Another design pattern that could be implemented in this system is the observer pattern. The pattern would be very useful anywhere that the system is trying to observe a state change. Most of the system sensors are monitoring a specific physical aspect of the system and changing an attribute’s value upon some physical state change. When the attribute value changes, the system should respond. This is where an observer would be beneficial by notifying the various dependent classes of the change. This would improve the efficiency of the system by ensuring that classes that need to be notified of changes will be notified as soon as the change occurs.

The decorator pattern analysis could be useful within the Login class. The system could have different levels of users with access to different UI functions. Using the decorator pattern, the system could be designed to change the functions available in the UI at runtime one the user has logged in. A user decorated as a service technician may have access to functions within the UI to modify the system function, while a user decorated as an operator may only be able to see the sensor display page. This would improve the usability of the system by ensuring that each user sees only what they need to see to get their job done. It would also improve the integrity of the system by ensuring that only those users who should have access to advanced features are granted that access.

The builder pattern analysis would be useful in creating the different displayed pages classes. Each page consists of many different elements that could become tedious and overwhelming if called individually. Using a builder to call these pages would reduce the complexity of the interfacing code, improving maintainability. It could also increase the reusability of the code by making it simpler to create new pages.

The strategy pattern analysis would be useful in the Hardware IO class. If an object is detected by the vision system, the Hardware IO could select an algorithm to scan the RFID and stop the conveyor motion. Otherwise it might select an algorithm that would keep the conveyor moving and keep the RFID scanner off. This would improve the efficiency of the system by reducing the amount of power used by the scanner and conveyor.

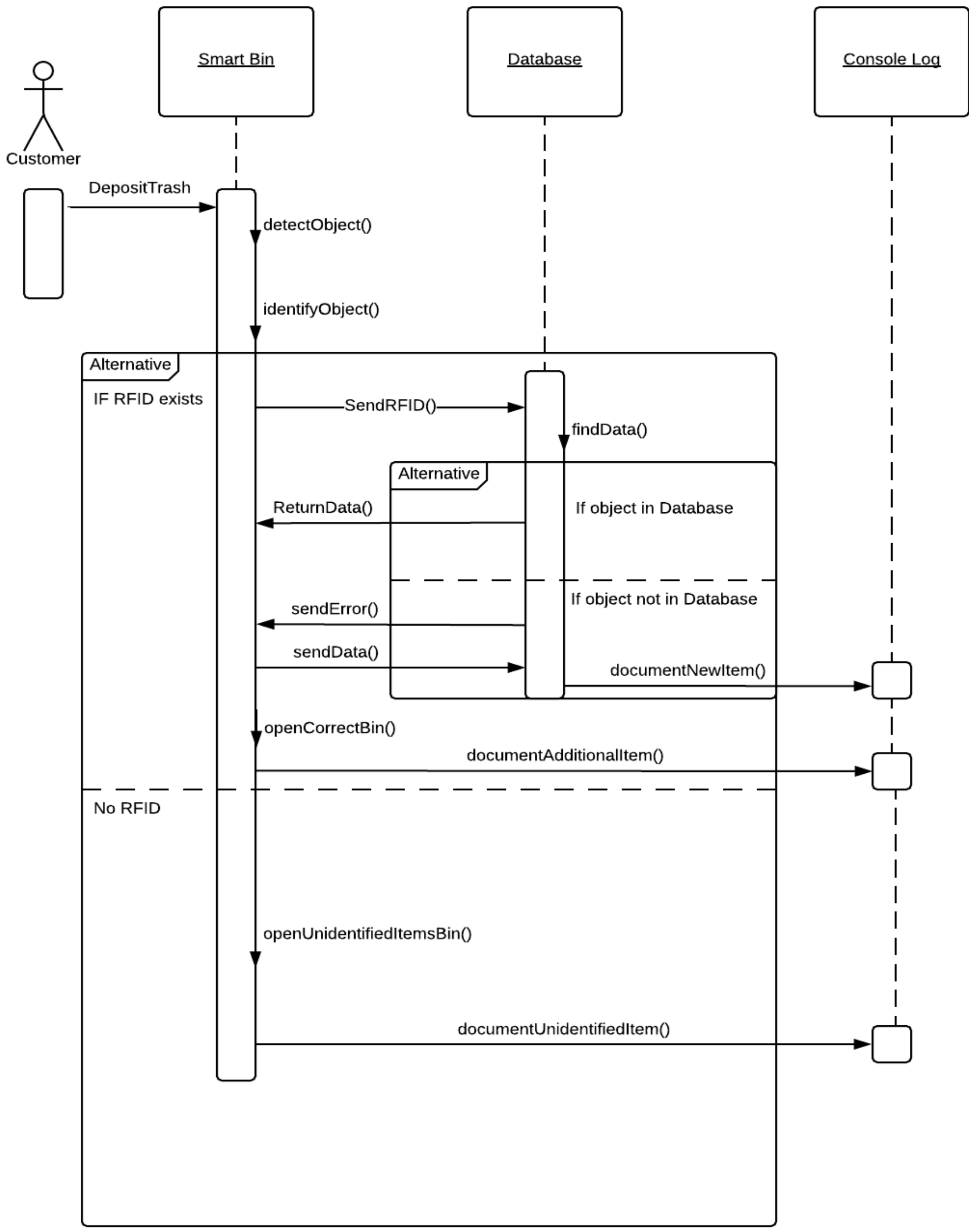
5. Sequence Diagram

5.1 Sequence Diagram For Use Case 2: Recycling Assistant - Individual Pre-Sorting

The design for our second use case was a machine that would be placed in airports or restaurants where customers could deposit their recyclable trash that would then be sorted into the appropriate bins.

The customer would place the item into a receptacle that would determine if the item included an RFID chip that designated it was a recyclable object. If so, it would send and receive data from the database to determine what bin to place the item into. When an item is added to a bin, a notification is passed to the console to update the amount of items that are currently held in that bin so that the technicians know when to empty the bins. If no RFID is recognized, then the bin would deposit the item into a nondescript bin that contains any items that are unidentifiable.

Figure 5.1a - Sequence Diagram For Use Case 2: Recycling Assistant - Individual Pre-Sorting



6. Interface Contracts and Exceptions

6.1 Interface Identification

The following is a list of interfaces for the system, and their contracts, that have been identified. Though incomplete it is representative of the interfaces typical of the system.

Diagnostics Interface:

The diagnostics interface is to be used as the focal point for system diagnostics. This interface will list any error states the system is currently in along with a description of the state. A user of this interface will also be able to browse logs from the system, this will be useful in determining past error states of the system as well as other key system attributes such as warnings and thresholds. The diagnostics interface will allow the user to view overall system diagnostics , as well as performing and viewing diagnostics for specific parts of the system.

Prerequisites for use of this interface are the system to be operational and the Server to be functional and in communication (via the Hardware IO) with the rest of the system. This interface takes an optional component ID as a parameter (to view diagnostics on that particular piece of the system). The interface returns the diagnostic information described earlier.

Reporting Interface:

The reporting interface will allow the user of the interface to query the system with preprogrammed queries that provide statistics from various past timeframes. This information can be used to gather data about processed recycling percentages, system performance, and ROI of the system. The reporting interface will allow the user to view overall system reports, as well as pulling reports that are specific to parts of the system, such as reports on historical sensor data for a particular sensor.

Prerequisites for use of this interface are the system to be operational and the Server to be functional and in communication with the Database Server. This interface takes an option report ID as a parameter (to select a specific preprogrammed report). The interface returns the report information described earlier.

Sensor Interface:

The sensor interface will allow the user of the interface to view details from the various sensors in the system. It will provide the user the ability to select a specific sensor and query the data for that sensor (as well a providing a means to access the reporting interface for that specific sensor for historical data). Example sensors include the capacity, load, and position sensors.

Prerequisites for use of this interface are the system to be operational and the Server to be functional and in communication (via the Hardware IO) with the sensors. This interface takes an option sensor ID as a parameter. The interface returns the sensor information described earlier.

Database Interface:

The database interface will allow the user of the interface to view, add, update, and delete entries to the recyclable waste database. This database contains the list of scannable ids that the system utilizes to sort recyclable waste into its various categories. Through this interface a user will be able to add a new id to the list of existing ids tied to a category, or add a new category all together (for example a new type of recyclable plastic).

Prerequisites for use of this interface are the system to be operational and the Server to be functional and in communication with the Database Server. This interface takes several optional parameters which vary whether the user is performing a view, add, update, or delete query to the interface. The interface returns the report information described earlier.

External Web Application Interface:

The external web application interface will allow residents of the community to view the status and amount of recycling that has been processed in a given time frame. Additionally it will show how much tonnage of waste has been saved from the land fill. This is available for outreach programs and to promote recycling use in the community.

Prerequisites for use of this interface are the database being in service and the server being operational and connected to the database. The interface will pull reports from the reporting interface as all of the queries are pre-determined with no ability to create custom queries from this interface. The interfaces returns the report information described earlier.

6.2 Potential Exceptions

The following are a list of several potential exceptions that were identified as likely to occur. Below they are documented along with their proposed handling. Though incomplete it is representative of the potential exceptions typical to the system.

Potential Exception #1:

When the user attempts to manually add an RFID code to the database it could be invalid.

Potential Exception #1 Handling:

- If the entered RFID code does not check out as a valid code an exception would be thrown and the handler would redirect the user back to the entry page.

Potential Exception #2:

The user could attempt to log in with an incorrect username or password.

Potential Exception #2 Handling:

- The handler would catch this and redirect back to the login function up to 10 times
- The handler would track how many incorrect logins were attempted in a given time period.
- After 10 incorrect login attempts a lockout would be activated

Potential Exception #3:

Any of the sensors could send an incorrect signal back to the IO. It could be a signal that was out of the range the IO was looking for or it could be a signal that does not make any sense to the hardware IO.

Potential Exception #3 Handling:

- Handler would catch this and would throw an error and alarm event notifying the diagnostics display to display an error indicating a potentially bad sensor
- The alarm would have data indicating what type of sensor it was and where it was located. The error that was thrown would also have an error code associated with it that indicated the sensor error and its location.

Potential Exception #4:

The conveyor motion module could be given a command to reverse the conveyor when it would be problematic to do so. For instance, if the reverse command was given to the recyclables belt and the mixed trash/recyclables belt was still set to forward, or if the gate was closed and recyclables were on the belt.

Potential Exception #4 Handling:

- The handler would throw an error notifying the user that the belt cannot be reversed while the gate is closed and trash/recyclables are present.
- The handler would throw an error notifying the user that the recycling belt cannot be reversed while the commingled belt is in the forward position and recyclables are present on the recycling belt.

Potential Exception #5:

The weight of trash/recyclables on any of the belts could exceed the rated limit.

Potential Exception #5 Handling:

- A handler would catch this exception and throw an error and alarm indicating the weight limit exceeded and which belt had exceeded the limit.