**CS360**

**Software Requirements Specification (SRS) Document**

The document in this file is an annotated outline for specifying software requirements, adapted from the IEEE Guide to Software Requirements Specifications (Std 830-1993).

**CS360**

**Habegger, Parker, Rasler, Shin, Park**

**Egg Alert And Real-time Logistics**

**Software Requirements Specification**

**Document**

**Version: (3.2)** **Date: (3/28/2012)**

**Table of Contents**

1. Introduction 5

1.1 Purpose 5

1.2 Scope 5

1.3 Definitions, Acronyms, and Abbreviations. 5

1.4 References 5

1.5 Overview 6

2. The Overall Description 6

2.1 Product Perspective 6

2.1.1 System Interfaces 6

2.1.2 Interfaces 6

2.1.3 Hardware Interfaces 7

2.1.4 Software Interfaces 7

2.1.5 Communications Interfaces 7

2.1.6 Memory Constraints 8

2.1.7 Operations 8

2.1.8 Site Adaptation Requirements 8

2.2 Product Functions 9

2.3 User Characteristics 9

2.4 Constraints 10

2.5 Assumptions and Dependencies 10

2.6 Apportioning of Requirements. 10

3. Specific Requirements 10

3.1 External Interfaces 10

3.2 Functions 11

3.3 Performance Requirements 13

3.4 Logical Database Requirements 13

3.5 Design Constraints 14

3.5.1 Standards Compliance 14

3.6 Software System Attributes 14

3.6.1 Reliability 14

3.6.2 Availability 14

3.6.3 Security 14

3.6.4 Maintainability 15

3.6.5 Portability 15

3.6.6 Usability 15

3.7 Organizing the Specific Requirements 17

3.7.1 Requirements Organized by Application Architecture 17

3.8 Additional Comments 19

4. Change Management Process 20

5. Document Approvals 22

6. Supporting Information 23

6.1. Product Overview 23

# 1. Introduction

The following subsections of the Software Requirements Specifications (SRS) outline the purpose of this document in relation to the product specified: The Egg Alert and Real-time Logistics (EARL) System.

## 1.1 Purpose

The purpose of this Software Requirements Specifications document is to list and detail the requirements inherent in the construction and maintenance of the Egg Alert and Real-time Logistics System. It is intended for the use of the client to verify that all required specifications for the EARL have been listed and considered. Further, this SRS will be used by the development team to ensure that all required design parameters are incorporated into the final product.

## 1.2 Scope

Software product to be produced: Software aspect of the Egg Alert and Real-time Logistics.

The goal of this project is the creation of a system that automates the process of determining when and where a chicken egg flow problem (egg jam) occurs on a system of conveyors through the chicken egg packaging process. Mechanical units will be installed along separate conveyors to track the flow of eggs down that specific conveyor; these units will report to a software program designed to determine if the flow is normal or abnormal. In the case of abnormal flow, the system will alert the user in real-time as to which specific line the problem has occurred on. In a typical poultry operation, thousands of feet of conveyor lines would need to be searched manually to locate a jam. This system would minimize the searching, thus reducing the labor cost needed to fix the problem. The system also eliminates unnecessary loss in performance by alerting users even when the packaging system is not in use.

## 1.3 Definitions, Acronyms, and Abbreviations.

EARL: Egg Alert and Real-time Logistics system

UART: Universal Asynchronous Receiver/Transmitter

BS2: BASIC stamp 2

CAT-5: Serial twisted pair category 5 RJ45 style cabling

CMap: Concept map

## 1.4 References

Voice of Customer Document

Functional Requirements Document

Extended Application Architecture

Group CMap Web Link: <http://cmapspublic.ihmc.us/rid=1K1K884SC-X1XFD0-2TWB/RaslerSoftEngCmap.cmap>

## 1.5 Overview

This document is segmented in such a way that the scope of the primary sections 1-2 is most suitable for those not involved in the construction of the EARL. The primary section 3 is most suitable for those involved in the construction of the software.

# 2. The Overall Description

## 2.1 Product Perspective

This product is being created for an environment with the following already established requisites: A production line with individual conveyors feed eggs from lines of chicken coops into primary lines, which in turn feed into the packaging area. Further, a Windows PC with touch screen exists at the user-operated packaging area.

A software system that operates a robotic sorter is already installed at the site. The sorter uses optical recognition and grading to evaluate the sizes of eggs on the conveyor lines and removes tose that are too large for the packing process. The mechanized solution elaborated upon in this document will be installed on the same PC and work alongside this system.

### 2.1.1 System Interfaces

As mentioned, this software system is to be operated on a PC which also operates a system that detects egg size and removes eggs too large for packing. The two systems’ GUIs will share the existing touch screen monitor, but otherwise will not interact; menu options to access alarms indicated by the EARL will be incorporated into the GUI, along with options for viewing system status and (possibly) log files.

Likewise, the sensor system must be integrated into the existing conveyor system. Sensor arrays will be installed according to specifications made by the client.

### 2.1.2 Interfaces

A touch-screen interface currently exists for the egg-sorting software system. The EARL will necessarily be integrated into this interface. The interface includes a GUI for user input and display. It should allow parameters to be easily adjusted and possibly provide a means of viewing log files. Touch-screen input will drive the user-adjustable parameters, alert toggles, and so on.

### 2.1.3 Hardware Interfaces

Mechanical counter devices should be installed on the conveyor in such a way as to gauge the flow rate of that particular conveyor belt. These sensors communicate with a conveyor specific microcontroller (BS2), and, potentially using a UART to buffer communication via serial lines, these serial transmissions should be converted to a USB specified transmission. This USB transmission will interface with the installation PC which should interpret and drive the communication of the whole system.

### 2.1.4 Software Interfaces

#### 2.1.4.1 Software Interface of BS2:

The Sponsor specific microcontrollers require PBASIC to communicate serially with the UARTs.

#### 2.1.4.2 Software Interface of the primary program:

The primary program will handle communication to the UARTs. This level of communication should be programmed for the Windows Environment. The communication data should interface with software that knows the state of each conveyor, utilizing algorithms to determine if an alert or response is necessary. This level of the software interfaces with a GUI, making alerts as necessary and describing the status of the conveyors. The preexisting touch screen should offer the user the ability to modify internal settings of the software - specifically including the ability to modify sensitivity settings and alert settings.

### 2.1.5 Communications Interfaces

#### 2.1.5.1 Communication Interface for UART to primary program software:

The communication interface between the UARTs and the PC will be established after the purchase of the UARTs and the BASIC Stamps. The specific hardware will determine the protocol necessary and may come prepackaged with the chip. The data will be transmitted over Cat-5 serial cable.

#### 2.1.5.2 Communication Interface from UARTs to BS2:

These two components will communicate at the hardware level as specified by the microcontroller and UART chosen.

#### 2.1.5.3 Communication Interface from BS2 to conveyor-positioned egg detector:

These two components will communicate at the hardware level as specified by the microcontroller.

### 2.1.6 Memory Constraints

This system’s memory requirements will not exceed the memory allowance of the current system, considering also the requirements of the packing program.

### 2.1.7 Operations

The user will interact with a GUI located at a packaging station, responding to alerts from the program by noting the location from whence the alert originated and manually fixing any problems.

### 2.1.8 Site Adaptation Requirements

Site adaptations would include:

Mechanical counters installed at intervals along individual conveyor lines.

Housing boxes established to hold both the UART and the BS2.

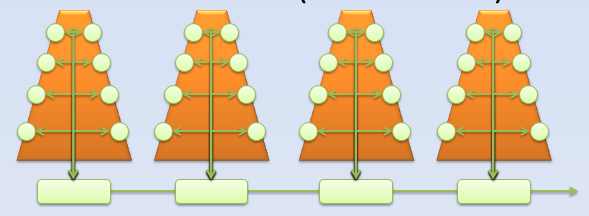
Wiring installed connecting counters to housing boxes.

Conduit installed to protect the wiring, prevent shorts, and prevent damage in the event of shorts.

Serial communication lines installed from each housing box to a workstation PC.

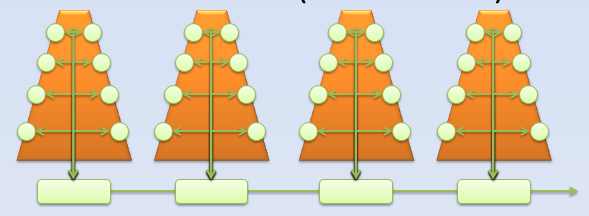
Note that, though these requirements are listed above, they are still considered in this document to be part of the overall system. They are listed above to aid in specifying a more generic system, rather than the specific instance that this document deals with.

## 2.2 Product Functions



Mechanical egg detectors (white circles) placed on conveyor lines (orange bands) detect the presence of eggs as they move down conveyor belts.

Egg Jam Here!



Alert!

Conveyance can stop due to mechanical issues or a system overload (egg-jam). Users are often remote and unaware of the problem. When eggs stop being detected on one line or at one location (red circles), and other detectors are detecting egg flow, then there is a jam.

Because eggs flow at differing rates, the software is responsible for determining what is a jam and what is not. It is also responsible for determining error situations. For example: if a module is unresponsive and other modules are responsive, then the module needs maintenance.

The program is also responsible for intelligibly displaying the status of the system. It should also provide methods for alerting that are expressive.

## 2.3 User Characteristics

The educational level, experience, and technical expertise required by the user is no more than required by methods already established in the process.

The user will be involved in the packaging process while operating this system, so considerations involving speed, usability and economy of (user) motion are important.

## 2.4 Constraints

The choice of hardware components will constrain methods used to communicate with the UARTs and BS2s. This is yet to be determined.

The system must be able to communicate across thousands of feet. This will limit the choice of cable and communication standards if signal repeaters are not an option.

The software will be hosted on a PC utilizing the Windows Operating System.

The GUI must fit the available screen dimensions (half of the screen being taken up by the packing program) and be legible at its current resolution.

## 2.5 Assumptions and Dependencies

The characteristic assumption of the system is that an egg jam can in fact be determined from the given amount of detectors installed.

If the development language requires modules to be installed on the PC (e.g. .Net or Java), these must be installed prior to software installation.

The response time of the system will be roughly proportional to the number of sensor units installed along the conveyor system, which is in turn dependent upon the length of the conveyor system and the chosen sensor density.

The frame rate of the GUI refresh will be dependent on the processor speed, the availability of a graphics card, and the choice of development language.

## 2.6 Apportioning of Requirements.

Not applicable.

# 3. Specific Requirements

## 3.1 External Interfaces

**3.1.1 Primary Application Interface**

3.1.1.1 Component Interfaces

The primary application should interface through RS485 connection, converting to serial communication at each module, possibly using a UART (provided by client) to buffer incoming communication. The primary application will request the current state and count of each individual module and receive them as they are polled in turn.

3.1.1.2 User Interfaces

The user should communicate through touch screen to adjust settings, control alerts, view logging information, etc.

**3.1.2 UART Interface**

The primary application should communicate with the UART (if used) acting as an intermediary to the communication line from the module (controlled by the microcontroller) to the primary application. This communication should be RS232, interfacing to a RS485 converter.

**3.1.3 Microcontroller Interface**

The microcontroller should interface with the primary application through the intermediary UART if it is incorporated, or directly to the main application through whatever means (e.g. RS232, RS242) is constrained by the provided components (TTL). If discovered, latency issues may force a new configuration.

## 3.2 Functions

**3.2.1. Presentation Layer**

FR1.1: Shall have a GUI

FR1.1.1: Shall have a Configuration Interface

FR1.1.1.1: Shall allow changing of alarm parameters

FR1.1.1.1.1: Shall allow changes to sample size

FR1.1.1.1.2: Shall allow changes to sample variance

FR1.1.1.1.3: Shall allow changes to alarm snooze length

FR1.1.1.1.4: Shall allow alarm sound file to be changed

FR1.1.1.1.5: Shall allow changes to sound volume

FR1.1.1.2: Shall allow changing of collector parameters

FR1.1.1.2.1: Shall allow changes to the number of sensor arrays displayed

FR1.1.1.2.2: Shall allow changes to the number of sensors per array displayed

FR1.1.1.3: Shall allow changing of serial communication parameters

FR1.1.1.3.1: Shall allow change to the serial port number

FR1.1.1.3.2: Shall allow changes to the timeout length

FR1.1.2: Shall have a Visual Alert

FR1.1.3: Shall have an Audio Alarm

FR1.1.4: Shall have a Statistics interface

FR1.1.4.1: Shall display the address of each module

FR1.1.4.2: Shall display the egg count for each sensor

FR1.1.4.3: Shall display the total egg count for each module

FR1.1.4.4: Shall display the total egg count

FR1.1.4.5: Shall allow changes to module address

FR1.1.5: May allow viewing of past log files

FR1.1.6: Shall have a Status interface

FR1.1.6.1: Shall show the status of each sensor

FR1.1.6.2: Shall display the status of each sensor array

FR1.1.6.3: Shall allow sensor arrays to be disabled

FR1.1.6.4: Shall allow the alarm to be disabled

FR1.1.7: Shall allow viewing of the log since activation

FR1.1.8: Shall allow disabling of changes

FR1.1.9: Shall require confirmation for reenabling changes

FR1.1.10: Shall allow the GUI to be closed and program shut down.

FR1.1.11: Shall allow the GUI to be resized

FR1.1.12: Shall allow the GUI to be minimized

FR1.1.13: Shall allow the GUI to be maximized

**3.2.2. Business Layer**

FR2.1: Shall run on Windows OS

FR2.1.1: Primary Application

FR2.1.1.1: Shall have a Bridge to Communication Driver

FR2.1.1.2: Shall have a Hardware Polling Process

FR2.1.1.3: Shall have a State Logic Process

FR2.1.1.4: Shall have a GUI Build/Update

FR2.1.2: Shall utilize a Serial Communication Driver

FR2.1.3: Shall utilize communication standards compatible with the microcontrollers

FR2.1.4: Shall utilize communication standards compatible with distance and sensor constraints

FR2.1.5: May incorporate error correction

FR2.2: Sensors (Microcontrollers)

FR2.2.1: Shall have a RS485 to RS232 Convertor

FR2.2.2: May buffer incoming communication

FR2.2.3: Shall utilize Communication Software

FR2.2.4: Shall have Counting Software

**3.2.3 Data Access Layer**

FR3.1: Primary Application

FR3.1.1: Shall be able to write log files

FR3.1.2: Shall be able to read log files

FR3.1.3: Shall have a Parser/Tokenizer

FR3.1.4: Shall be able to write configuration files

FR3.1.5: Shall be able to read configuration files

**3.2.4. Persistence Layer**

FR4.1: Primary Application

FR4.1.1: Shall record Configuration Settings

FR4.1.2: Shall maintain log files of system status changes

FR4.1.3: Shall have State Values

FR4.1.4: Shall maintain egg counts for each sensor

FR4.1.5: Shall maintain egg counts for the system as a whole

FR4.1.6: May store status changes in a database

FR4.1.7: May store egg counts in a database

FR4.2: Microcontroller

FR4.2.1: Shall maintain a unique address

FR4.2.2: Shall maintain state values

FR4.2.3: Shall store egg counts per sensor

**3.2.5. Site preparation**

3.2.5.1 Install hardware counters

3.2.5.2 Install communication lines

3.2.5.3 Install communication controllers

3.2.5.4 Integrate into already built workstation

## 3.3 Performance Requirements

The system will be installable on one PC. It is tailored to service one contiguous conveyor system, although theorhetically it could retooled to monitor several at once. As it responds to only one terminal, it is largely intended for a single user; the remote alert system is currently geared to one device, which is intended to be operated by a different user than the operator stationed at the terminal.

The sensor capacity is currently not known, but we expect to load it with between 10 and 20 units, according to the length of the conveyor system and the density of sensor units required to accurately represent the state of the system. The time to poll all sensors is proportional to the number of sensors installed, with approximately a maximum of 5 seconds spent per sensor before the relay times out and moves on to the next one. Jam determination will be scheduled at least once per 5 full-system polls, and should take less than 2 minutes. Alerts will be scheduled immediately, if necessary, and polling will be suspended until resumed by the operator. This is a separate process from suspending the alarm, which merely keeps it from sounding until the alert system is reset.

Logging will occur at the time of an alert and should take 10 seconds or less.

## 3.4 Logical Database Requirements

The data that is most likely to be stored in a database are the log files that describe the occrences of alert situations. A typical entry would include the date, the time, a snapshot of the system state (perhaps captured as a binary string), the system’s determination of the problem (jam or faulty module), and a confirmation or correction by the operator (optional). This data could be in a future system version that would use machine learning to better identify problem locations. The database would further be able to record a count of the day’s production. Use would be on an as-needed basis, so availability should be high. The database can be accessed from the touch-screen interface for further review and printing. Data will be retained at the discretion of the operator.

## 3.5 Design Constraints

The system currently installed is based around a Windows PC. As our software will be installed onto the same PC, it must be compatible with the PC’s operating system. The GUI must fit the available screen space and should be easily operable at the given resolution.

The conveyor system extends for thousands of feet throughout the farm. Signal repeaters are currently deemed cost-prohibitive. This will limit the choice of available communication hardware and standards.

### 3.5.1 Standards Compliance

There are no standards we currently know to apply to this project. The log files generated by the software will be in a format to be determined at a future date.

## 3.6 Software System Attributes

### 3.6.1 Reliability

The EARL should be able to identify a jam situation within 8 minutes from the initiation of stoppage at least 95% of the time. False positives (e.g. a particularly long break in egg production interpreted as a stoppage) should be limited to less than one occurrence per operation.

### 3.6.2 Availability

It is necessary for the EARL to be available on demand, as the encapsulating conveyor system is run on a variable schedule. It should be able to operate daily for a minimum of 18 hours continuously. In the event of a system failure, the EARL should be recoverable within 5 minutes.

### 3.6.3 Security

The function of the EARL is self-contained. The only interaction between the EARL and other programs is that its GUI is displayed concurrently on the same monitor as the GUI for the egg-sizing robot that is operated by the PC. The only trading of data occurs during the writing of log files. Should the EARL crash, it will not affect the operation of other programs. Further, should there be a malfunction or failure of a system module, the EARL will be able to report the occurrence and identify the bad microcontroller or UART.

### 3.6.4 Maintainability

The function of the EARL is divided into the following modules:

* GUI: Touch-screen interface for locating jams, disabling alarms, and viewing logs.
* Stamp programming: software for the BASIC microcontrollers.
* Data communication: software to relay messages between the PC and the sensor system.
* Jam logic: determination of a jam or other error from the collected input signals.
* Alert system: initiates visual and audio warnings to the user interfaces.
* Logging system: records time and locations of jams.

Commentary explaining system functions shall be incorporated into the code base. A change log making more detailed explanations may be implemented.

### 3.6.5 Portability

Portability is not a strong consideration for the EARL. The software is being tailored to integrate with a legacy system using components that the client already owns or is familiar with. Known portability issues are as follows:

* Software for the UARTs and BASIC stamp units will be coded in Parallax BASIC. This code is specific to the chosen equipment and may require retooling if these components should become unavailable.
* The main program (signal interpretation, jam determination and location, alerts, and user interface) may be coded in .NET, Java, or some other language with a strong visual component. While the intended platform is a Windows-based PC, this part could potentially be ported to other operating systems. We will not explore this possibility in the scope of this project.
* The communications software, which will be based largely on Windows API calls, may have hardware-specific components or parameters.

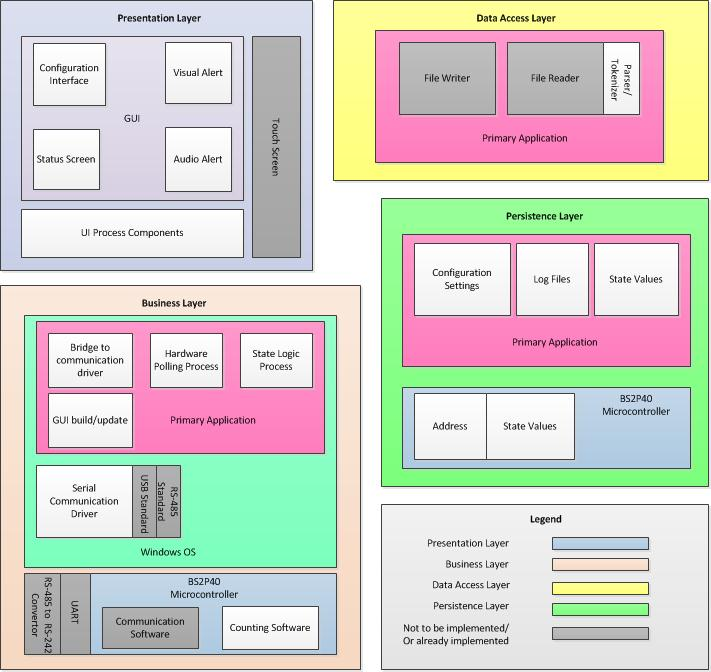
### 3.6.6 Usability

The EARL is most likely to be used by nontechnical personnel. As such, the user interfaces will stress ease of use. The GUI should be minimalistic and menu-driven, featuring components that are oversized and easy to read.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Characteristic** | **H/M/L** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **A** | **B** | **C** |
| 1 | Correctness | M | X | 1 | 1 | 1 | 5 | 1 | 1 | 8 | 1 | 1 | B | C |
| 2 | Efficiency | L | 1 | X | 3 | 2 | 5 | 6 | 2 | 8 | 2 | A | B | C |
| 3 | Flexibility | M | 1 | 3 | X | 3 | 5 | 3 | 3 | 8 | 3 | 3 | B | C |
| 4 | Integrity/Security | L | 1 | 2 | 3 | X | 5 | 6 | 4 | 8 | 9 | A | B | C |
| 5 | Interoperability | H | 5 | 5 | 5 | 5 | X | 5 | 5 | 8 | 5 | 5 | B | C |
| 6 | Maintainability | M | 1 | 6 | 3 | 6 | 5 | X | 6 | 8 | 6 | 6 | B | C |
| 7 | Portability | L | 1 | 2 | 3 | 4 | 5 | 6 | X | 8 | 9 | A | B | C |
| 8 | Reliability | H | 8 | 8 | 8 | 8 | 8 | 8 | 8 | X | 8 | 8 | B | C |
| 9 | Reusability | L | 1 | 2 | 3 | 9 | 5 | 6 | 9 | 8 | X | A | B | C |
| A | Testability | M | 1 | A | 3 | A | 5 | 6 | A | 8 | A | X | B | C |
| B | Usability | H | B | B | B | B | B | B | B | B | B | B | X | C |
| C | Availability | H | C | C | C | C | C | C | C | C | C | C | C | X |

## 3.7 Organizing the Specific Requirements

### 3.7.1 Requirements Organized by Application Architecture



Each logical section will create a hierarchical node to nest the requirements as such:

FR0:Egg Alert and Real Time Logistics

FR1: Presentation Layer

FR1.1: Shall have a GUI

FR1.1.1: Shall have a Configuration Interface

FR1.1.1.1: Shall allow changing of alarm parameters

FR1.1.1.1.1: Shall allow changes to sample size

FR1.1.1.1.2: Shall allow changes to sample variance

FR1.1.1.1.3: Shall allow changes to alarm snooze length

FR1.1.1.1.4: Shall allow alarm sound file to be changed

FR1.1.1.1.5: Shall allow changes to sound volume

FR1.1.1.2: Shall allow changing of collector parameters

FR1.1.1.2.1: Shall allow changes to the number of sensor arrays displayed

FR1.1.1.2.2: Shall allow changes to the number of sensors per array displayed

FR1.1.1.3: Shall allow changing of serial communication parameters

FR1.1.1.3.1: Shall allow change to the serial port number

FR1.1.1.3.2: Shall allow changes to the timeout length

FR1.1.2: Shall have a Visual Alert

FR1.1.3: Shall have an Audio Alarm

FR1.1.4: Shall have a Statistics interface

FR1.1.4.1: Shall display the address of each module

FR1.1.4.2: Shall display the egg count for each sensor

FR1.1.4.3: Shall display the total egg count for each module

FR1.1.4.4: Shall display the total egg count

FR1.1.4.5: Shall allow changes to module address

FR1.1.5: May allow viewing of past log files

FR1.1.6: Shall have a Status interface

FR1.1.6.1: Shall show the status of each sensor

FR1.1.6.2: Shall display the status of each sensor array

FR1.1.6.3: Shall allow sensor arrays to be disabled

FR1.1.6.4: Shall allow the alarm to be disabled

FR1.1.7: Shall allow viewing of the log since activation

FR1.1.8: Shall allow disabling of changes

FR1.1.9: Shall require confirmation for reenabling changes

FR1.1.10: Shall allow the GUI to be closed and program shut down.

FR1.1.11: Shall allow the GUI to be resized

FR1.1.12: Shall allow the GUI to be minimized

FR1.1.13: Shall allow the GUI to be maximized

FR1.2: Shall work with UI Process Components

FR1.3: Shall utilize touch screen interface

FR2: Business Layer

FR2.1: Shall run on Windows OS

FR2.1.1: Primary Application

FR2.1.1.1: Shall have a Bridge to Communication Driver

FR2.1.1.2: Shall have a Hardware Polling Process

FR2.1.1.3: Shall have a State Logic Process

FR2.1.1.4: Shall have a GUI Build/Update

FR2.1.2: Shall utilize a Serial Communication Driver

FR2.1.3: Shall utilize communication standards compatible with the microcontrollers

FR2.1.4: Shall utilize communication standards compatible with distance and sensor constraints

FR2.1.5: May incorporate error correction

FR2.2: Sensors (Microcontrollers)

FR2.2.1: Shall have a RS485 to RS232 Convertor

FR2.2.2: May buffer incoming communication

FR2.2.3: Shall utilize Communication Software

FR2.2.4: Shall have Counting Software

FR3: Data Access Layer

FR3.1: Primary Application

FR3.1.1: Shall be able to write log files

FR3.1.2: Shall be able to read log files

FR3.1.3: Shall have a Parser/Tokenizer

FR3.1.4: Shall be able to write configuration files

FR3.1.5: Shall be able to read configuration files

FR4: Persistence Layer

FR4.1: Primary Application

FR4.1.1: Shall record Configuration Settings

FR4.1.2: Shall maintain log files of system status changes

FR4.1.3: Shall have State Values

FR4.1.4: Shall maintain egg counts for each sensor

FR4.1.5: Shall maintain egg counts for the system as a whole

FR4.1.6: May store status changes in a database

FR4.1.7: May store egg counts in a database

FR4.2: Microcontroller

FR4.2.1: Shall maintain a unique address

FR4.2.2: Shall maintain state values

FR4.2.3: Shall store egg counts per sensor

## 3.8 Additional Comments

The Axiomatic Design Process was used to map individual Functional Requirements to Design Parameters. These mappings were used to isolate instances of redundancies. The Design Matrix is shown below:

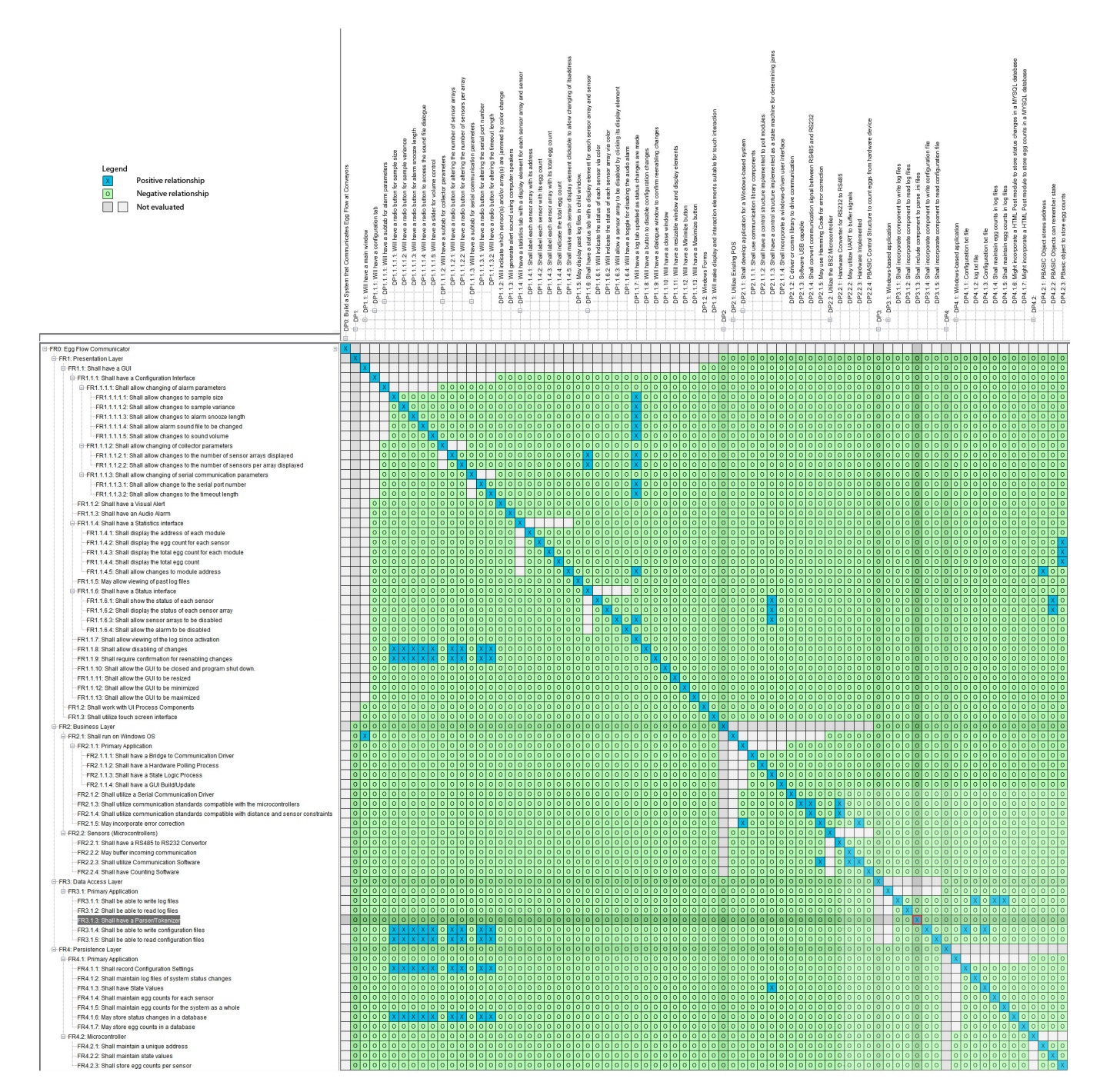


Figure 1: EARL Design Matrix

# Change Management Process

As requirements change, the Client Interface will communicate these changes to the team, and a team decision will be made whether the requirements change will be sufficient enough to create new tools including but not limited to: a new SRS document, a new Application Architecture, new FMEA reports, a new design matrix, etc…. The Axiomatic Design Software tool Acclaro will help streamline this process.

Simple changes can effectively be considered using the Axiomatic Design Tool, with any residual effects observable from the streamlined creation of design views from the tool. Any changes to requirements should be formalized in a new SRS, and a team consensus should be sought before any changes are officially negotiated.

As objects of the project are elaborated, they will be preemptively validated with the Sponsor, allowing for low-overhead resolution of changes prior to development. A new Voice of Customer document should also be created for quality assurance and reference.

All documents will be created with an iterative version number, allowing for a paper trail of decisions. Prior documents will be maintained for this sake.

# Document Approvals

|  |  |  |
| --- | --- | --- |
| **NAME** | **DATE** | **SIGNATURE** |
| **Matthew Rasler\_\_\_\_\_** | **\_\_\_\_\_\_\_\_\_\_\_\_\_** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Andrew Habegger\_\_\_** | **\_\_\_\_\_\_\_\_\_\_\_\_\_** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Mark Parker\_\_\_\_\_\_\_** | **\_\_\_\_\_\_\_\_\_\_\_\_\_** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Eun Young Shin\_\_\_\_** | **\_\_\_\_\_\_\_\_\_\_\_\_\_** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Sunyoung Park \_\_\_\_** | **\_\_\_\_\_\_\_\_\_\_\_\_\_** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

# Supporting Information

## 6.1. Product Overview

The goal of this project is the creation of a system that automates the process of determining when and where a chicken egg flow problem (egg jam) occurs on a system of conveyors through the chicken egg packaging process. Mechanical units will be installed along separate conveyors to track the flow of eggs down that specific conveyor; these units will report to a software program designed to determine if the flow is normal or abnormal. In the case of abnormal flow, the system will alert the user in real-time as to which specific line the problem has occurred on. In a typical poultry operation, thousands of feet of conveyor lines would need to be searched manually to locate a jam. This system would minimize the searching, thus reducing the labor cost needed to fix the problem. The system also eliminates unnecessary loss in performance by alerting users even when the packaging system is not in use.