Software Requirements Specification (SRS)

Project Automate Paint Defect Analysis

Authors: Lisa Doan, Alex Besinger, Patrick McCormick, Samantha Oldenburg, Clayton Peters

Customer: General Motors

Instructor: James Daly

1 Introduction

This Software Requirements Specification document outlines Group 8's solution to the client's need to automate the recording of paint defects on vehicles during production. The topics covered in this document include: an introduction to the client's problem, the team's proposed solution to address the problem such as how the product system should function, specific system and client requirements, and diagrams illustrating use cases and scenarios when the system would be used. Additionally, a prototype of the proposed system with instructions on how to run it and sample scenarios is also provided in the latter portion of the document. At the end of the document, external sources that were referenced appear in a list.

1.1 Purpose

The Software Requirements Specification (SRS) document serves to outline the technical requirements needed by the developers working on creating the system for automating the collection of automotive paint defect analysis. This document provides specific insight such as requirements, constraints, and detailed explanations of the proposed system to combat the client's problem. The intended audience for this document are the developers working on creating the system for the client. The developers and technology experts make sure that system is "technically and economically feasible" (Pfleeger & Atlee, page 147).

1.2 Scope

To put the proposed system in perspective, the result of the project will be a system with a web interface that allows the client analysts to input defect data about certain models of cars in a standardized fashion. The system will automate the recording of paint defects on vehicles during production and then allow users to run reports on the data. The system will eliminate the need for paper diagrams and will allow automatically generate the desired reports from the entered data. Additional benefits include a decrease

in resources (i.e. ink, paper, etc.) and human error that comes from physically transcribing the vehicle paint defects. Users of the system will be able to track defects over time.

The application domain of the system will be part of the client's company's existing computer infrastructure. The application can be run on a standard desktop or laptop computer with internet connection which are located at each inspection checkpoint in order to produce reports identical to those currently generated. The application will support all vehicles in production at any of the three plants, such as the Lansing Delta Township, Lansing Grand River, and Lake Orion Assembly, where the application will be utilized.

The front-end provides a user interface which will present the user with a wireframe of the vehicle in question, on which they can select the location, type, and severity of defects. The defects are then stored in a database, which can be queried with reports. Using the information inputted into the forms on the web application, a variety of reports can be generated including: weekly charts that contain a summary of the defects week by week over a given time range, quality assurance reports which summarizes vehicle inspections over a given time range, and a summary of analysis report which provides a numeric summary of defects over a selected date and shift. The software will not be able to determine what is and what is not a defect nor will it fix the problems on the vehicle.

1.3 Definitions, acronyms, and abbreviations

To clarify the unique terminology, acronyms, and abbreviations used in the document, the list below clearly specifies the meanings of these terms.

- **SRS (Software Requirements Specification)**: A document of the software system to be developed that discusses the functional and nonfunctional requirements. It also includes use cases that describe how users (client analysts) would interact with the software.
- **Defect**: An impurity, scratch, or other imperfection in the paint on a car. Analysts are trained to recognize all the different types of defects.
- **Reports**: A summary of information gathered from inspections of cars that allow the resulting data on defects to be viewed in an easy to read manner. Three types of reports. All three types of reports can be specified to only include the defects on a specified vehicle or specified vehicle part. In addition, all reports can be specific to a given time range.
    - The **weekly chart** is an excel file a user can create that contains a summary of the defects week by week over a given time range.
    - The **QA report** summarizes vehicle inspections over a given time range. The report contains numerous tables and figures to easily visualize paint defects. The first table displays a vehicle wireframe with the aggregate result of all the inspections and the defects found. Each defect displayed on the vehicle wireframe is color-coded to a type of defect found on the vehicle, displayed in a legend. The following tables display numerical data about defects specific to various locations on the vehicle. The last table in

the QA report displays a pie chart representing each type of defect in regard to the total number of defects.
    - The **summary of analysis reports** is a numeric summary of defects over a selected date and shift. These reports contain the number of total defects, the number of units inspected, and the number of defects per unit. The defect information is displayed by location as well as defect type. The total list of defects used to generate the report is listed at the end of the report with information about the color, model, location, defect type, and defect notes stored in the database.
- **DPU** is Defects Per Unit, which is defined as the average number of defects on multiple vehicles that were inspected.
- **HTTP** refers to the HyperText Transfer Protocol.
- **Analyst**: A user of the system who is trained to identify the different types of paint defects that can be present on a car. Analysts are the users who will enter defect data while working on the factory floor.

1.4 Organization

Following this introductory section, the rest of the Software Requirements Specification document will further explain the specifics of the software system being produced. Section 2 focuses on the context for needing the proposed system, how it should function, and the related constraints. Section 3 provides a detailed listing of the requirements specific to the system following a hierarchical numbering scheme. Section 4 addresses the modeling requirements which includes use cases diagrams, a state diagram, and a high-level class diagram. Section 5 discusses the proposed prototype and the details of the system with instructions on how to run it as well as sample scenarios. Section 6 contains a list of references used throughout the document. Finally, Section 7 provides contact information used for further inquiries.

2 Overall Description

This section will cover general information about the project perspective, functions, and various requirements and constraints. More detailed descriptions of the project requirements and functions will be covered in latter sections.

2.1 Product Perspective

This project is designed to be used by employees of several automotive plants to record and track defects in the paint of cars on an assembly line. There are several constraints the system will need to operate under. First, the end user system will need to run on a Windows computer, while the backend will need to run on appropriate server hardware. The end user system will need to present a user interface that allows a user to indicate the location, severity, and type of defects. In addition, users will be able to query server data by running reports. The frontend and backend will communicate using HTTP requests to both send and receive data from the database.

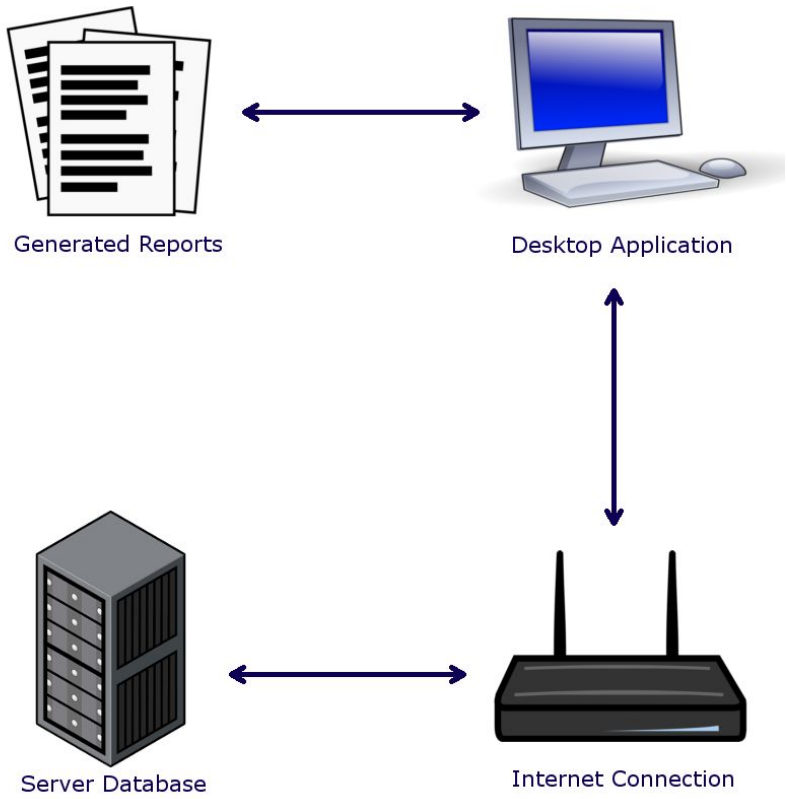

Fig 1. *Pictorial representation of the bigger system*

2.2 Product Functions

The end user system will initially need to present a user interface where they can select the factory they are working at, the checkpoint they are inspecting at, and the type of car being inspected. This information is stored on the server alongside the defect data that the user inputs later.

Second, another menu will allow users to indicate the location, severity, and type of defects. This menu will display a wireframe for the selected model of car, that allows users to place the defect in the appropriate location on the vehicle.

Lastly, end users will be able to run three types of reports over previously stored defect data. The first type of report is a QA report, which summarizes vehicle inspections over a specific time range, which is chosen by the user when running the report. It contains all the data stored for the defects over that period, the total number of cars inspected, and the defects per unit. It also contains several tables that show an aggregated car diagram with locations of all recorded defects, and a pie chart that displays the ratios of all defect types over this period.

The second report is a Summary of Analysis report, which is a numeric summary of defects over a given period that were entered by a specific analyst. The user running the report is able to specify the date range to run the report on. This report contains the shift the analyst was on, the check point being inspected, and a summary of the number of defects, number of units, and number of defects per unit.

The final report type is a weekly chart, which outputs an Excel file containing weekly data over several months. The number of months to run the report on is selected by the user. This report contains weekly statistics for each facility, part name, machine, and data for all defects during that period.

2.3 User Characteristics

There will be two primary types of users for this program. The first type will be technicians that are running paint inspections in the car factories. These technicians will be trained to identify types of paint defects, and to run a complete inspection of the entire paint job on a car. After identifying a paint defect of a particular type, these technicians will then enter the defect into the software to store its information in the database. The second type of user will be an administrative user that will utilize the reports the system produces.

2.4 Constraints

The system will have two separate sets of hardware constraints, one for the end user program, and one for the server architecture. The end user system will need to run on a Windows computer, while the backend will need to run on appropriate server hardware. Both setups require an internet connection, and the server architecture requires enough memory to store the database of past defects. The exact amount of storage required will depend on how long the customer wishes to store past information for.

2.5 Assumptions and Dependencies

It is assumed that the end user program will be run on a Windows Vista or newer Windows system. It is also assumed that the technician using the end user program will have been trained to recognize all the types of paint defects, as it is not the software's purpose to explain how to qualify defects, instead it only stores data on the defects.

2.6 Proportioning of Requirements

A user directory system is a feature that could be included in a future release. A user directory could store what factory a user works in, what shifts they work, and further information for users. Currently, any user can access any factory, and input defects for any shift, checkpoint, and model. A user directory system could be used to verify that a user is entering data for an appropriate shift, and reduce the possibility of mistakes when entering data.

Another possible feature that could be added in the future is the release of another standalone program explicitly for use by administrative agents running reports. Currently, the same executable will run on both factory computers and administrative employee computers. In theory, this means that an administrator could enter defects from their office computer, while a factory worker could run reports from the factory checkpoint.

Dividing these features into two executables would simplify work for both types of employees, while hiding unnecessary function.

3 Specific Requirements

1. Analyst loads application and enters identification information
    1.1. Analyst enters their first and last name
    1.2. Analyst enters the start and end time of their working period, along with which shift they are working
    1.3. Analyst selects which checkpoint vehicles are inspected
    1.4. Analyst saves information entered and information is stored in a database
2. Analyst inspects a vehicle (repeat until end of shift as vehicles are inspected)
    2.1. Analyst selects model of car inspected
        2.1.1. Wireframes of selected model shown
    2.2. Analyst selects type of defect
    2.3. Analyst marks area of defect on the vehicle wireframes shown
        2.3.1. Analyst will drag an ellipsoid over the area of defect and shape the ellipsoid to accurately represent the defect
        2.3.2. Color of mark will depend on the type of defect selected in step 2.2
    2.4. Analyst enters any notes about defect
    2.5. Analyst saves information entered and information is stored in a database
    2.6. If necessary, the analyst is able to remove a vehicle inspection information if they made a mistake entering information
3. Analyst selects type of report to generate
    3.1. Analyst selects vehicle inspections to include in report
        3.1.1. Able to select or deselect vehicles to include based on GM facility, checkpoint at which the vehicle was inspected, time and date of inspection, or type of defect
            3.1.1.1. Data from at least ten vehicle inspections must be included to generate a report
    3.2. Data regarding selected vehicle inspections are retrieved from database and collated
    3.3. Selected report is generated

4 Modeling Requirements

This section provides use case, class, sequence, and state diagrams that address the specifications of the requirements defined in section 3.

1. **Use case diagrams** illustrate possible situations where the user interacts with the system.

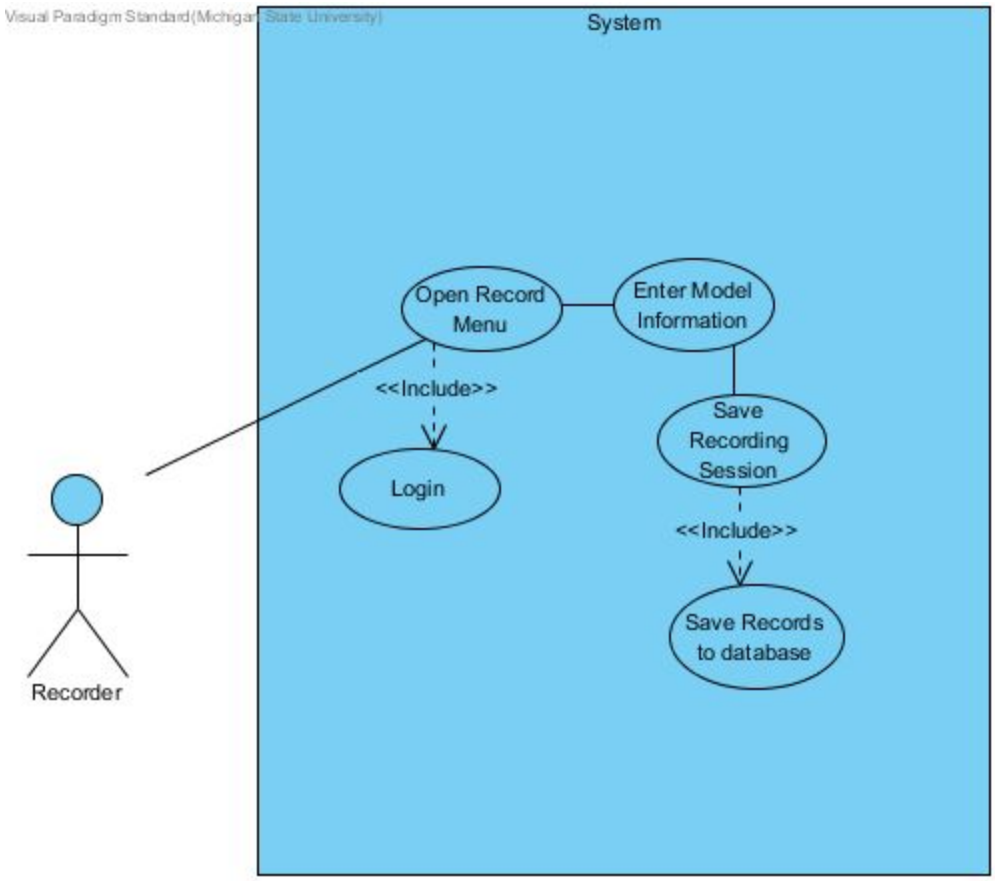

Fig 2. *Use Case Diagrams*

- Analyst loads application and enters identification information

Use Case:	Login
Actors:	Analyst
Description:	Analyst enters identifying information, selects their work period, and selects the checkpoint where vehicles are to be inspected.

- Analyst inspects a vehicle (repeat until end of shift as vehicles are inspected)

Use Case:	Record Defects
Actors:	Analyst
Description:	Analyst selects model of vehicle from given

	wireframes, mark the defects on the wireframe by drawing ellipsoids over the areas, and enters note about the defect. Information is saved and stored in a database.

- Analyst removes defect record

Use Case:	Remove Defect Record
Actors:	Analyst
Description:	Analyst can remove a defect record if there is an error.

- Analyst selects type of report to generate

Use Case:	Generate Report
Actors:	Analyst
Description:	Analyst selects vehicle inspections to include in the report by specifying the GM facility, the checkpoint of inspection, time of inspection, or type of defect.

2. **Class diagram**

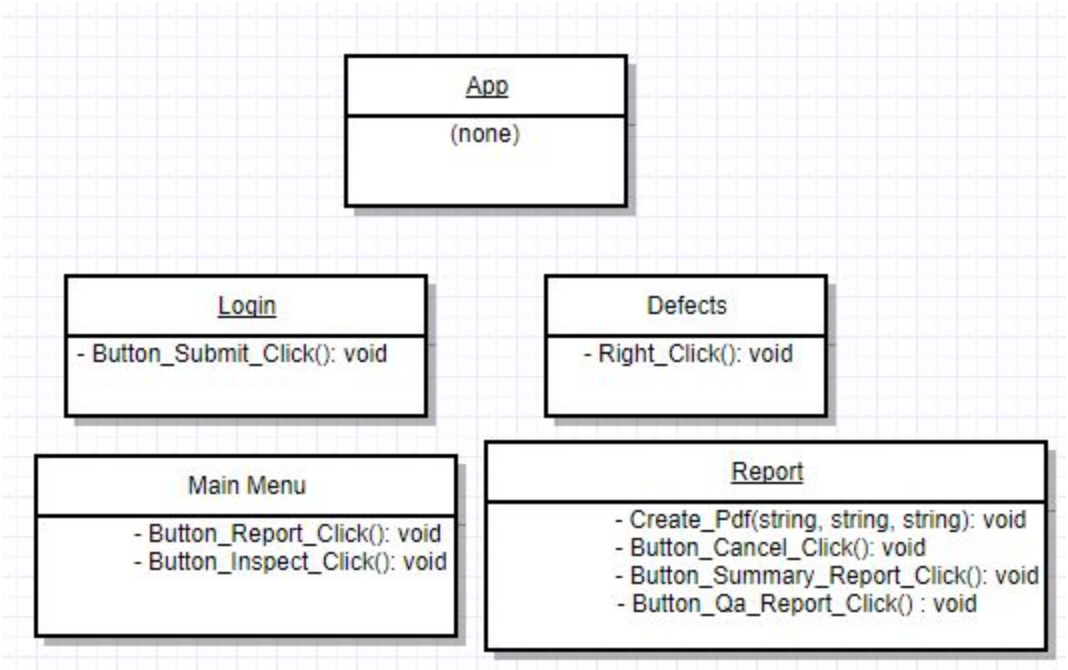

Fig 3. *Class Diagram*

3. **Sequence diagram**

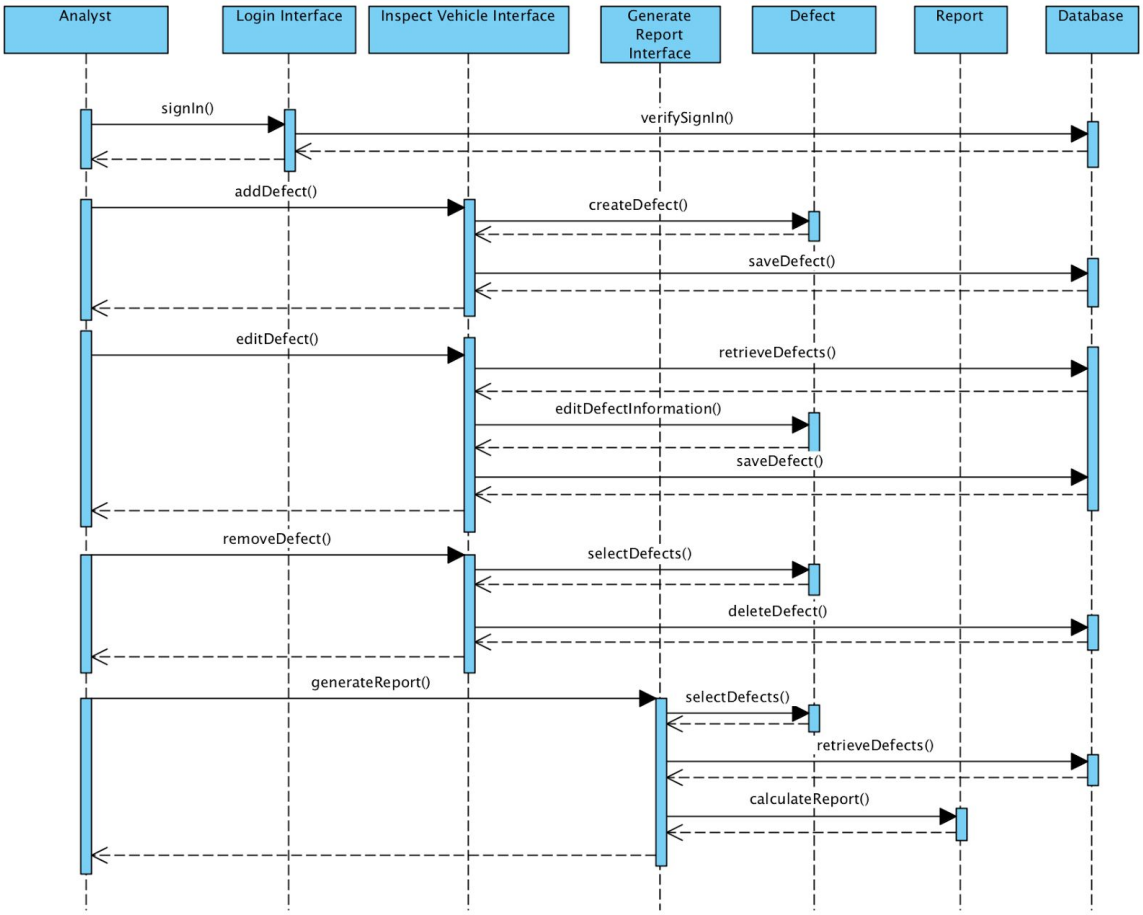

Fig 4. *Sequence Diagram*

4. **Data dictionary** for back-end system
    a. This describes the attributes, operations, relationships, and UML extensions for each of the classes in the class diagram.

Element Name		**Description**
Defects	Provides interface for viewing a given vehicle and the defects associated with it. Supports 3D model of the vehicle with color-coordinated defect visualization.	
Attributes		

	components:System.ComponentModel.IContainer	Required designer variable
Operations		
	Right_Click(obj, RoutedEventArgs): void	Handles the right click event
Relationships	One of the 4 UI Windows of the program	
UML Ext.	(none)	

Main_Menu	When user has logged in, class that provides options to generate reports or inspect further vehicles	
Attributes		
	(none)	
Operations		
	Button_Report_Click(obj, RoutedEventArgs): void	Handles the generate report button click, takes user to Report windows
	Button_Inspect_Click(obj, RoutedEventArgs): void	Handles the inspect vehicles button click, takes user to Defects window
Relationships	One of the 4 UI Windows of the program, main menu takes user to other UI windows	
UML Ext.	(none)	

Login	Class handling the actual login process interacting with the SQL database	

Attributes		
	(none)	
Operations		
	Button_Submit_Click(obj, RoutedEventArgs): void	Handles the button click for a user to login
	Process_Login(obj)	Handles the actual login interacting with the SQL database, displays whether successful or not
Relationships	One of the 4 UI Windows of the program, allows user to login with username/password and retrieve everything he or she has access to (vehicles, reports, etc.)	
UML Ext.	(none)	

Reports	Class handling report generation (PDF creation)	
Attributes		
	(none)	
Operations		
	Create_Pdf(string, string, string): void	Handles actual PDF creation and saving PDF to directory on Windows system
	Button_Cancel_Click(obj, RoutedEventArgs): void	Handles when user clicks cancel button in UI
	Button_Weekly_Report_ Click(obj, RoutedEventArgs): void	Handles the create pdf button click and calls Create_Pdf() to create the pdf file with strings user has provided in textboxes

	Button_Summary_Report(obj, RoutedEventArgs): void	Handles the create pdf button click and calls Create_Pdf() to create a summary of analysis report
	Button_Qa_Report_Click(obj, RoutedEventArgs): void	Handles the create pdf button click and calls Create_Pdf() to create a QA report
Relationships	One of the 4 UI Windows of the program, allows user to generate reports	
UML Ext.	(none)	

5. **State diagram**

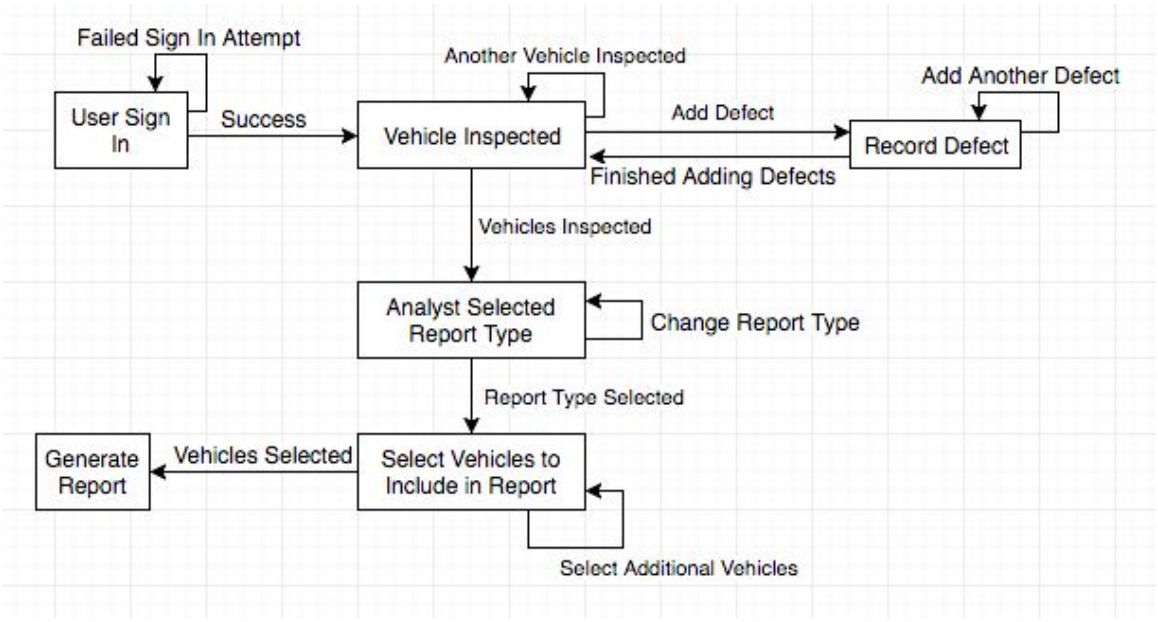

Fig 5. *State Diagram*

5 Prototype

There will be an interface, the recording interface, solely dedicated to marking down paint defects that the reporter observes. There are three main components to this interface: the canvas, the toolbox, and the menu. The menu will hold miscellaneous functionality such as saving the data inputted, exiting the recording interface, and switching which car model the user is working with. The canvas will display an image of

the wireframe of the car model the user is working on. Using their mouse, the user can create ellipsoids that mark an area where there was paint defect, change the dimensions of the ellipsoid, change the color of the ellipsoid, and erase ellipsoids they have already made. The toolbox allows the user to modify what action they will perform by clicking and/or dragging the mouse on the canvas. There will be a toggle button for erasing ellipsoids, and one which will allow the user to create new ellipsoids. There will also be a toggle button to allow a user to change the dimensions of an ellipsoid when they click-and-drag the ellipsoid. Only one of these three toggle buttons can be toggled at a time, and the one which is toggled will determine which action the user can do on the canvas. There will be a swatch full of colors. Each color correlates to a type of defect. One color is selected at any time, and will be the color newly created ellipsoids will be.

There will be an interface to generate reports. The interface will allow you to filter by date (using a calendar interface), model of car, type of defect, location of defect, and the plant the information was originally recorded. All filters other than the date filter will implemented by a group of checkboxes per filter field, with one checkbox for each possible option (e.g. GMC Acadia, Chevrolet Traverse, and Buick Enclave will each have their own checkbox for the plant filter).

There will be a main menu interface. This interface will have several buttons, each allowing the user to move to a different interface such as the recording interface (e.g. there will be a button in the menu interface which will open the recording interface when clicked). It will be possible to access any interface from the main menu interface (directly or indirectly). When a user logs in, they are brought to the main menu interface. When a user closes an interface directly accessed from the main menu interface, they are brought back to the main menu interface. The user may also log out using the main menu interface.

There will be a login interface. The login interface is the initial state of the system, and the system goes back to that state when a user logs out. A user must login to access the main menu interface, and therefore all other functionality in the system. A user logs in using a unique four-digit code assigned to them by their supervisor.

These interfaces make up the front-end system. The front-end system runs from a terminal in each factory. These systems will connect to a single back-end system which hosts the recorded information (see Section 4 above).

5.1 How to Run Prototype

The system is a desktop application that requires a computer running Windows Vista SP2 or higher version of Windows operating system. The computer must have Visual C++ Redistributable 2010, as well as .NET Framework 4.0 installed.

Prototype V1 will have little functionality, and its main purpose will be to display the user interface as well as to show our interpretation of the requirements.

5.2 Sample Scenarios

An employee, Bob, was given the unique four-digit-code 7924 by their supervisor, Chris. This is their login code for the system until further noticed by their

supervisor. Chris tells Bob to analyze the next 20 Camaros coming out of the factory line for paint defects. Bob moves to the terminal running the system.

Prototype V1 has little functionality, and its main purpose is to display the user interface as well as to show our interpretation of the requirements

6 References

[1] Pfleeger, Shari Lawrence, and Joanne M. Atlee. "Software Engineering: Theory and Practice," Pearson Higher Education, New Jersey, 2010.

[2] CSE 435 Group 8. Michigan State University, 2017, http://cse.msu.edu/~oldenbu6/cse435.

7 Point of Contact

For further information regarding this document and project, please contact **Prof. J. Daly** at Michigan State University (dalyjame at cse.msu.edu). All materials in this document have been sanitized for proprietary data. The students and the instructor gratefully acknowledge the participation of our industrial collaborators.