Project Manager: Camille Williams
Team #2 Members: Marc King (TL), Aly Lakhani, Rodney Lewis, Jacob Vacheresse, Fan Zhang
Date Created: February 7, 2016. **Revised:** February 10, 2016

Ice Cream Manager

Software Project Management Plan

CIS 375 – Winter 2016 – Professor Steiner

Part 1 Due Date: February 11, 2015

1. Introduction

This Software Project Management Plan details the functionality, methodology, and estimates for the Ice Cream Manager software product. Ice Cream Manager is a product which allows the manager of an ice cream truck fleet to efficiently and easily manage their inventories, sales history, and truck routes.

1.1 Project Scope

Ice Cream Manager will be a graphical interface that will employ a system that will increase efficiency for the role of managing ice cream trucks. This application will allow the user to keep track of inventory levels, calculate sales though inventory levels, and manage their fleet of trucks. Inventory levels will be able to be displayed and modified for each truck. Each truck has an assigned route that may be manually edited by the user, or changed via a batch file. Additionally, each route may also have the zones which comprise it modified. If a truck is assigned to a route, the fuel efficiency of those trucks will be calculated and displayed. Trucks will have an assigned driver that will be able to report voting results of a given route. Shipments of ice cream will have an expiration date associated with it to prevent waste, older ice cream will be sold first or disposed of if the date passes. This software will support multiple languages.

Major Inputs

- Batch files for
    - Trucks
    - Inventory
    - Routes
    - Drivers
- Keyboard/mouse input for
    - Sales History
    - Trucks
    - Routes
    - Inventories

Processes

- Sales history
- Truck fuel history
- Customer voting
- Creating/Modifying
    - Trucks
    - Inventory
    - Routes
    - Drivers
- Seasonal profiles

Outputs

- Inventory displays modified by season
- Calculated sales history
- Calculated truck fuel history
- Results of customer voting

1.2 Major Software Functions

- Graphical User Interface (GUI)
- Sales History Calculations (SHC)
- Inventory Management (IVM)
- Route Management (ROM)
- Database Integration (DBI)
- Batch File Processing (BFP)
- Freshness and Waste (FAW)
- Internationalization (I18N)
- Fuel Tracking and History (FTH)
- Customer Product Voting (CPV)
- Seasonal Profiles (SSP)
- Labor Cost Tracking (LCT)

1.3 Performance/Behavior Issues

Not applicable. There are no performance or behavior issues that need to be accounted for during this project.

1.4 Management and Technical Constraints

- End of the semester imposes a dead drop date for this application.
- Due to the team's responsibility to other classes, time is a limited resource.
- Change will inevitably occur, so we have to be mindful of how in-depth our own functionalities go.
- This software will be developed using C#, .NET, and WinForms.
- The Model-View-Controller paradigm will be used when designing this software package.

2. Project Estimates

2.1 Historical Data Used for Estimates

Based on previous team member projects, the following is historical data used for some of the following estimates:

Program	Author	Lines of Code	Hours	Lines/ Hour	FPs
Conway's Game of Life	Marc	2500	30	84	24
Millikan Oil Drop Experiment Simulator	Marc	12300	200	62	162
Postfix Calculator	Jacob	150	5	30	
Game Character Creator	Jacob	500	11	46	

Table 2.1 - Historical Data

2.2 Estimation Techniques Applied and Results

The following section will detail the Wild Guess, Lines of Code, Function Point, and Task Driven estimates. The Wild Guess is exactly as it sounds; a wild guess made by the team regarding how long the project should take. The Lines of Code estimate is essentially another wild guess made regarding the number of lines of code each part of the project will require, which is then compared to historical data to calculate the time that will be required. The Function Point estimate examines the complexity and importance of the various functionality of the product, and uses those to calculate the time required. Finally, the Task Driven estimate looks at the detailed task breakdown of the project and assigns time estimates to each task; those time estimates are then totaled.

2.2.1 Estimation Techniques

The following subsections will detail the numbers involved in each of the estimations.

2.2.1.1 Wild Guess Estimation Technique

Based on historical data and gut feeling, we have decided upon a wild guess of 650 hours. That value consists of 500 hours for the team, and 150 hours for the project manager. The team's estimate can be broken down into 50 hours for Part 1, 300 hours for Part 2, and 150 hours for Part 3. The project manager's estimate can be broken down into 100 hours total for all three parts of the project plan, 25 hours for weekly progress reports, and 25 hours for plan meetings.

Wild Guess Estimate = 650 hours

2.2.1.2 Lines of Code Estimation Technique

Function	Estimated LOC
Graphical User Interface (GUI)	3200
Sales History Calculations (SHC)	850
Inventory Management (IVM)	600
Route Management (ROM)	400
Database Integration (DBI)	300
Batch File Processing (BFP)	600
Freshness and Waste (FAW)	350
Internationalization (I18N)	500
Fuel Tracking and History (FTH)	350
Customer Product Voting (CPV)	350
Seasonal Profiles (SSP)	350
Labor Cost Tracking (LCT)	350
Total estimated lines of code	**8200**

Table 2.2.1.2 Lines of Code Estimations

Based on our historical data, we average 56 lines of code (LOC) per hour. However, none of our historical data is derived from projects that involve the same level of planning as this project. Because of that, we will use a standard rate of 20 LOC/HR (provided in lecture) to calculate this estimate.

LOC Estimate = Total LOC / LOC Rate
LOC Estimate = (8200 LOC) / (20 LOC/HR)
LOC Estimate = 410 hours

2.2.1.3 Function Point Estimation Technique

See the Appendix for a breakdown of the Information Domain counts.

Weighting Factor					
Information Domains	Count	Simple	Average	Complex	FP
External Inputs	12	3	4	**5** ✓	60
External Outputs	12	4	5	**7** ✓	84
External Inquiries	19	3	**4** ✓	6	76
Internal Logical Files	11	7	**10** ✓	15	110
External Interface Files	1	**5** ✓	7	10	5
				Total FP Count	**335**

Table 2.2.1.3.1 Initial estimate 1 –based on function point

		Importance Scale					
Number	**Question**	**0**	**1**	**2**	**3**	**4**	**5**
1	Does the system require reliable backup and recovery?	✓					
2	Are specialized data communications required to transfer information to or from the application?						✓
3	Are there distributed processing functions?	✓					
4	Is performance critical?				✓		
5	Will the system run in an existing, heavily utilized operational environment?				✓		
6	Does the system require online data entry?	✓					
7	Does the online data entry require the input transaction to be built over multiple screens or operations?	✓					
8	Are the ILFs updated online?	✓					
9	Are the inputs, outputs, files, or inquiries complex?					✓	
10	Is the internal processing complex?					✓	
11	Is the code designed to be reusable?		✓				
12	Are conversion and installation included in the design?	✓					

13	Is the system designed for multiple installations in different organizations?	✓					
14	Is the application designed to facilitate change and ease of use by the user?						✓

Table 2.2.1.3.2 FP Value Adjustment Factors

Total Value Adjustment Factors = 25

Total Function Points = Weighted Function Points × (0.65 + 0.01 × Total Value Adjustment Factors)
Total Function Points = 335 × (0.65 + 0.01 × 25)
Total Function Points = 302 FP

According to our historical data, we are able to average a Function Point Rate of 0.8 FP/HR.

Function Point Estimate = Total Function Points / Function Point Rate
Function Point Estimate = (302 FP) / (0.8 FP/HR)
Function Point Estimate = 378 hours

2.2.1.4 Task Driven Estimation Technique

Activity ⟶	**CC**	**Planning**	**Risk analysis**	**Engineering (hrs)**		**Construction release (hrs)**		**CE**	**Totals (hrs)**
Task →				**Analysis**	**Design**	**Code**	**Test**		
Function ↓									
GUI				10	10	10	5	n/a	35
SHC				5	5	5	2	n/a	17
IVM				5	5	5	2	n/a	17
ROM				5	5	5	2	n/a	17
DBI				10	15	5	5	n/a	35
BFP				5	5	5	2	n/a	17
FAW				5	5	5	2	n/a	17
I18N				5	5	8	4	n/a	22
FTH				5	5	5	2	n/a	17
CPV				5	5	5	2	n/a	17
SSP				5	5	5	2	n/a	17
LCT				5	5	5	2	n/a	17
Totals	**50**	**50**	**10**	**70**	**75**	**68**	**32**		**355**

% effort	14%	14%	3%	20%	21%	21%	19%		
Key: CC - customer communication, CE - customer evaluation, for Function acronym meanings see section 1.2									

Table 2.2.1.4 Task Driven Estimates

Task Driven Estimate = 355 hours

2.2.2 Estimates

2.2.2.1 Initial Estimation

1. Estimate based on Wild Guess: 650 hours (Team: 500, Project Manager: 150)
2. Estimate based on Lines of Code: 410 hours
3. Estimate based on Function Point: 378 hours
4. Estimate based on Tasks: 355 hours

2.3 Reconciled Estimate

The Wild Guess, Lines of Code, Function Point, and Task Driven estimates vary within reasonable expectations, so we will find the average of those four values to determine our reconciled estimate. Because the Wild Guess estimate is the only estimate that takes the project manager's time into account, we will calculate a weighted average giving the Wild Guess estimate twice as much importance as the other estimates.

Weighted Average = 0.4*Wild Guess + 0.2*Lines of Code + 0.2*Function Point + 0.2*Task Driven
Weighted Average = 0.4*650 + 0.2*410 + 0.2*378 + 0.2*355
Weighted Average = 489 hours

2.3.1 Initial Estimation

Estimation for Part 1: The only estimation techniques that take planning into account are the Wild Guess and Task Driven. Because they both consist of guesses for the planning, we have to use that for our initial estimation for Part 1. The Wild Guess and Task Driven estimates indicate that we will spend 50 hours on Part 1.

Estimation for the entire project: The weighted average estimation between Wild Guess, Lines of Code, Function Point, and Task Driven is 489 hours for the entire project.

2.3.2 Post-Part 1 Estimation

After working on Part 1, the total time spent will be 60 hours for the team, while we estimated that it would take the team 50 hours to complete Part 1. Because these values are not significantly off, we will not be revising our estimates based on the time spent on Part 1.

2.4 Project Resources

1. **People:** Marc King, Aly Lakhani, Rodney Lewis, Jacob Vacheresse, Fan Zhang, Camille Williams

2. **Hardware:**
    1. Windows-based desktop and notebook computers.
    2. Windows, Android, and iOS tablets and smartphones.

3. **Software:**
    1. Operating Systems: Windows 7, Windows 10, Elementary OS
    2. Development Environment: Visual Studio 2015
    3. Unit Testing: Visual Studio 2015
    4. Documentation: Microsoft Office 2013 & 2016
    5. UML Modeling: Visual Studio 2015
    6. Third-Party Libraries: SQLite Database

4. **Tools:**
    1. Software Configuration Management: GitHub
    2. Issue Tracking: GitHub
    3. Communication: Slack, SMS, Email, Google Hangouts
    4. Document Sharing: Slack, Google Drive

3. Risk Management

3.1 Project Risks

- **Requirement addition:** Addition or changes to the product requirements by the customer.
- **Customer ambiguity:** Differences between what the customer wants and what the team thinks the customer wants.
- **Team ambiguity:** Differences in specification and requirement interpretations among team members resulting in components not working together.
- **Incomplete planning:** Project is not planned or designed completely or correctly when coding begins.
- **Planning estimation:** Underestimation in size and duration of planning phase.
- **Coding estimation:** Underestimation in size and duration of coding phase delaying testing.
- **Testing estimation:** Underestimation in size and duration of testing phase resulting in a poorly tested product.
- **Software Training:** Variation in skills and experience with software among team members causing time needed to learn tools.
- **Poor communication:** Lack of, or unclear communication between team members.
- **Scheduling problems:** Scheduling conflicts among team members due to school or work.

- **Low Productivity:** Pressure of other classes might take the focus off of this project.
- **Absent team member:** One or more team members unable to work on project because of illness or personal problems.
- **Team member conflict:** Significant conflict or argument between team members.
- **Data loss:** Loss of documents, code or other data.
- **Unpredicted risks:** Risks not predicted that do not have management plans.

3.2 Risk Table

Risk	Prob.	Impact	Mitigation Plan	Contingency Plan
Requirement Addition	High	High	Ask customer about changes every class. Estimate extra time for additions.	Use extra time to implement additions. Divide tasks among team members.
Customer Ambiguity	Medium	Critical	Communicate with customer and ask about specifications. Show customer project before deadline.	Meet to plan how to fix project before deadline. Clarify what customer wants.
Team Ambiguity	Low	Medium	Share documents in progress on google docs and code on GitHub. Review others work.	Team member assigned to component revises it. Possibly distribute revision among other team members.
Incomplete Planning	Low	High	Plan extensively. Add extra time to coding estimation.	Assign plan modifications to team member. Redistribute previous tasks.
Planning Estimate	Medium	High	Estimate high. Set early team deadlines.	Redistribute unfinished tasks and alter schedule as needed. Team meets to finish near-deadline tasks.
Coding Estimate	High	Medium	Estimate high. Set early team deadlines. Plan effectively.	
Testing Estimate	High	Critical	Estimate high. Set early team deadlines. Unit test during coding.	
Software Training	High	Medium	Distribute tasks by skills. Use technology that team members are confident with.	Team members will help others and answer questions. Redistribute tasks if needed.
Poor Comm.	Low	Medium	Team uses Slack app, phone and email. Meetings expected every Tuesday/Thursday.	Set mandatory team meetings after class Tuesday/Thursday.
Scheduling Problems	Medium	Low	Posted availability. Team will communicate changes to availability in slack. Early deadlines.	Reschedule or reassign tasks.
Low Productivity	Medium	Medium	Maintain proactive, positive team attitudes. Keep early team deadlines.	Redistribute tasks among team members. Meet to finish tasks close to deadline.
Absent Team Member	Low	Medium	Communicate need for absence with team on Slack. Early deadlines.	Reassign tasks. Change availability and schedule based on need.
Team Conflict	Low	Low	Team is friendly and professional.	Resolve conflicts in person.
Data Loss	Low	Critical	Documents kept on Google Docs or uploaded. Code automatically saved and uploaded on GitHub.	Rewrite code or documents.

Unpredicted	Low	Low	Update table with new predicted risks.	Communicate new problems. Find solution based on problem. Update table.

Table 3.2 Risk Management

3.3 Overview of Risk Mitigation, Monitoring, Management

Throughout the project the team will follow the risk mitigation plans. The team lead will set early deadlines to keep the team on schedule. The project manager will monitor risks and the effectiveness of our mitigation plans. In general, contingency plans will be used when risks arise. One team member will update the risk table as new risks are identified.

4. Project Schedule

4.1 Project Task Set

A. Software Project Management Plan Part 1 - Scope
B. Software Project Management Plan Part 1 - Planning
C. Software Project Management Plan Part 1 - Estimates
D. Software Project Management Plan Part 2 - Revision
E. Software Project Management Plan Part 3 - Revision
F. Requirements Gathering
G. Creation of Requirements Document
H. Traceability Analysis
I. Traceability Matrix
J. Entity-Relationship Diagrams
K. Use Case Diagrams
L. Activity Diagrams
M. Sequence Diagrams
N. Class Diagrams
O. Base Functionality Data Model
P. Additional Functionality Data Model
Q. Base Functionality View
R. Additional Functionality View
S. Base Functionality Controller
T. Additional Functionality Controller
U. Data Model Testing
V. User Interface Testing
W. Deployment

4.2 Functional Decomposition

- View Sales History
    - Query Sales from DB
    - Display Sales History
        - Display Trucks
        - Display Routes
        - Display Zones
- View Waste History
    - Query Waste from DB
    - Display Waste History
        - Display Route
        - Display Items
        - Display Inventory
- View Fuel History
    - Query Fuel from DB
    - Display Fuel History
        - Display Truck
- View Labor Costs History
    - Query Labor Cost from DB
    - Display Labor Costs History
        - Display Driver
        - Display Route
- View Voting History
    - Query Voting from DB
    - Display Voting History
        - Display Zones
        - Display Voted Upon Items
- Modify Route
    - Query Route from DB
    - Query Zones from DB
    - Display Route
        - Display Zones
    - Update Route in DB
- Modify Truck
    - Query Truck from DB
    - Query Items from DB
    - Query Drivers from DB
    - Display Trucks
        - Display Driver
        - Display Fuel
        - Display Route
        - Display Inventory
    - Update Truck in DB

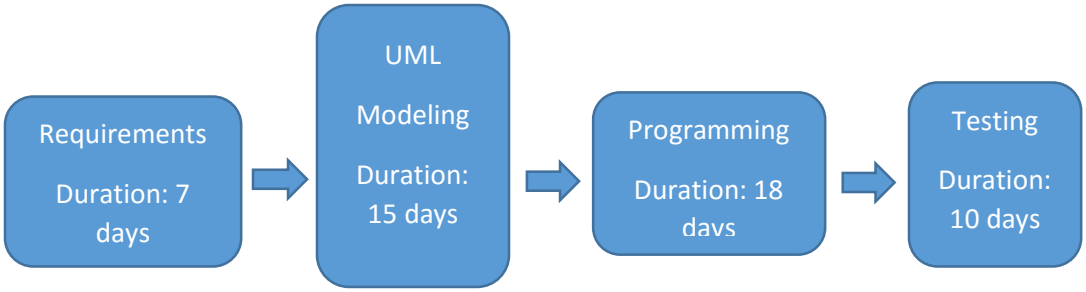
- Modify City
    - Query City from DB
    - Query Zones from DB
    - Display City
    - Update City in DB
- Modify Driver
    - Query Driver from DB
    - Display Driver
    - Update Driver in DB
- Modify Profile
    - Query Profile from DB
    - Display Profile
    - Update Profile in DB
- Modify Settings
    - Display Settings Form
        - Change Language
    - Update Settings File
    - Redraw Affected Views
- Batch Process Sales
    - Read Batch Sales File
    - Parse Batch Sales File
    - Update Sales in DB
- Batch Process Route
    - Read Batch Route File
    - Parse Batch Route File
    - Update Routes in DB
- Batch Process Inventory
    - Read Batch Inventory File
    - Parse Batch Inventory File
    - Update Inventories in DB

4.3 Task network

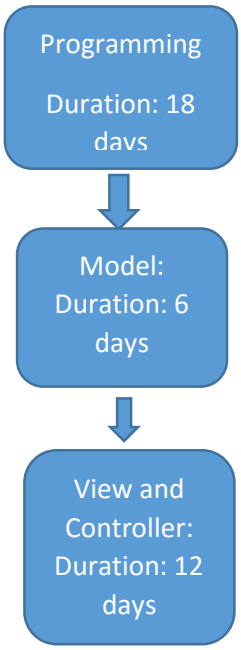
Using Section 4.1:

Task	Duration	Predecessor	Successor	Deliverable	Completion
A	2 days	-	B	High-level description of software project.	100%
B	2 days	A	C	Task-level breakdown of software project.	100%
C	3 days	B	F	Four software estimation techniques.	100%
D	7 days	C	E	Revised SPMP.	0%
E	7 days	D	-	Revised SPMP.	0%
F	3 days	C	G	Rough requirements list.	0%

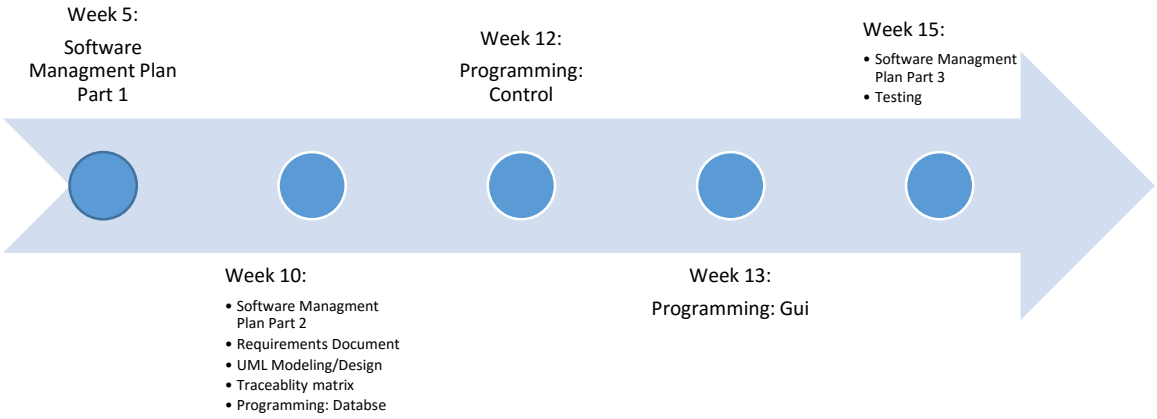
G	4 days	F	H	Formal requirements document.	0%
H	3 days	G	I	List of traceable artifacts.	0%
I	4 days	H	J	Traceability matrix.	0%
J	3 days	I	K	Entity-Relationship diagrams.	0%
K	4 days	J	L	Use Case diagrams.	0%
L	2 days	K	M	Activity diagrams.	0%
M	2 days	L	N	Sequence diagrams.	0%
N	4 days	J,M	O	Class diagrams.	0%
O	3 days	N	P,Q,S	Base functionality database tables and queries.	0%
P	3 days	O	R,T	Additional functionality database tables and queries.	0%
Q	4 days	O	R,V	Base functionality user interface.	0%
R	4 days	P,Q	V	Additional functionality user interface.	0%
S	2 days	O	T	Base functionality controller methods.	0%
T	2 days	P,S	U,V	Additional functionality controller methods.	0%
U	5 days	T	-	Robust data model.	0%
V	5 days	R,T	-	Deployable product.	0%
W	0 days	V	-	Deployed product.	0%



Dependencies:



4.4 Timeline chart



5. Staff Organization

5.1 Team Structure

Name	Role	Task Focus	Contact Information
Camille Williams	Project Manager	Management, project scheduling	camillwi@umich.edu 313-978-5568
Marc King	Team Lead	Facilitating communication, high-level planning, user interface programming, logic programming.	mjking@umich.edu 248-238-4542
Aly Lakhani	Developer	User interface programming, task estimations, quality assurance.	amlakhan@umich.edu 248-302-8063
Rodney Lewis	Developer	UML modeling, logic programming, documentation.	rodneyll@umich.edu 313-903-1010
Jacob Vacheresse	Developer	Algorithm implementation, quality assurance, logic programming.	jvachere@umich.edu 313-655-7833
Fan Zhang	Developer	Quality assurance, data modeling, data programming.	fzfan@umich.edu 917-573-8550

Table 5.1 – Team Structure

5.2 Management Reporting and Communication

The following list details the constraints and requirements related to communication among the team, and reporting to the project manager:

1. **Management Reporting.** Weekly progress reports will be provided by the team lead to the project manager on Saturdays by 6:00 PM EST. These reports will highlight the work that was accomplished during the previous week, which tasks are going to be worked on during the following week, and any issues that have arisen that affect the project. The team lead will create these reports by summarizing information that is provided by all of the team members.

2. **Communication.** The primary method of intra-team communication will be through the online collaboration tool Slack. This tool will allow varied discussions related to the project to be available in a single location. Team members are expected to periodically check the Slack platform for communications and to reply to such messages within 24 hours. Communication between the team and the project manager will be Slack, email, and/or text messages. Team members are expected to responds to such messages within 24 hours as well.

3. **Meetings.** In addition to internet communication, a minimum of one weekly, in-person meeting will be held following lecture on Thursday evenings from 8:00 PM through 10:00 PM. Video conferences and other in-person meetings may be scheduled in advance to address project needs.

6. Tracking and Control Mechanisms

6.1 Quality assurance and control

To assure the quality of our software project, we will be using a very specific workflow. The first step of this workflow is to identify a requirement, since everything should stem from a requirement. If such a requirement does not exist, then no changes should be made. After the requirement is identified, a branch of the source code repository on GitHub will be created specifically to make the change or addition related to that requirement. All development related to this change will be made within the branch that was created. The name of the branch should clearly identify what change or addition it contains.

When development on a branch is proceeding, the first step will be to create one or more unit tests for the change or addition. Only after the unit tests have been devised, and created, will actual development of the change or addition proceed.

After a developer considers their change or addition to be complete, and it continues to pass all unit tests, then they will submit a pull request to have the team lead, and any other developers closely tied to that requirement, examine and comment upon their changes. If the changes are found to pass all unit tests, adhere precisely to the requirement, and follow the coding standards and guidelines, then it will be merged with the master branch. The largest team requirement for this workflow is that the master branch must always compile without warnings, and all unit tests must continue to pass.

If any issues arise unexpected, they will be submitted using the GitHub issue tracking system. There the issue will be tracked and commented upon until the problem source code is located and corrected. The GitHub issue tracking system will also allow us to analyze and view reports related to our software issues. Such analyzation may allow us to identify a particularly issue-prone piece of source code and rewrite or refactor to stop it from continuing to generate issues.

All source code will be marked with a requirement identifier to enforce compliance. This identifier will be in the form of REQ####, where #### is a unique number associated with that requirement in the requirements document. These unique identifiers for the requirements will assist us in tracing requirements through all documents and source code. If any piece of source code cannot be associated with a requirement, it will not be merged with the master branch.

After the software is considered complete, extensive testing will be manually performed by the team. Team members will focus on aspects of the software for which they had minimal responsibility.

6.2 Change Management and Control

To manage and track changes to the artifacts related to this project, we will be using Google Docs for miscellaneous documentation and GitHub for source code and UML modeling documents. Both of these systems will allow us to audit changes made to documentation and source code. Changes to documentation will occur without oversight, due to the ability to easily undo any incorrect changes. Changes to source code will occur as detailed in Section 6.1 of this document. If a change to a requirement is proposed, that requirement will be traced through all the SCM artifacts to measure and document the impact of that change.

When working with source code within a branch (see Section 6.1 for more information) all commits will be very specific changes or additions, and the commit message will be clear and succinct. The commit message should include why the change is necessary, how it addresses the issue, any potential side effect the change may have, and the requirement identifier related to the change. Using this process, any changes or additions can be easily undone without impacting later changes, and requirements can be traced through our commit history.

7. Appendix

Information Domains
The following Information Domains were used during the Function Point Estimate in Section 2.2.1.3

External Inputs (EIs)
1. Sales Data Batch File
2. Route Data Batch File
3. Inventory Data Batch File
4. Modify Route UI
5. Modify Truck UI
6. Modify City UI
7. Modify Driver UI
8. Modify Profile UI
9. Modify Zone UI
10. Modify Inventory UI
11. Modify Items UI
12. Change Settings UI

External Outputs (EOs)
1. Sales History
2. Inventory History
3. Trucks
4. Routes
5. Cities
6. Zones
7. Waste History
8. Fuel History
9. Voting History
10. Profiles
11. Labor Costs
12. Drivers

External Inquiries (EQs)
1. View Sales History
2. View Inventory
3. View Routes
4. View Trucks
5. View Cities
6. View Zones
7. View Waste
8. View Items
9. View Fuel
10. View Voting
11. View Profiles

12. View Labor Costs
13. View Drivers
14. Modify Inventory
15. Modify Routes
16. Modify Trucks
17. Modify Items
18. Modify Profiles
19. Modify Drivers

Internal Logical Files (ILFs)

1. Inventory DB Table
2. Route DB Table
3. Truck DB Table
4. City DB Table
5. Zone DB Table
6. Item DB Table
7. Voting DB Table
8. Profile DB Table
9. Driver DB Table
10. Fuel DB Table
11. Waste DB Table

External Interface Files (EIFs)

1. Settings File