Engineering Change Management Tool: Senior Project

By

Janel Gumbs

Abdoul Razakou Hamissou

Elisa Miller

Arth Suthar

Abstract

The Engineering Change Management Software (ECMS) process is categorized into five following steps:

Identify need for engineering change

Identify the need for engineering change will be trigger by an issue(**input**), This process and activity is controlled by organizational procedure. The mechanisms/users include issue initiator, initiator's manager. An issue could be described as a product related concern with a potential of creating a change in the form, fit or function. The **output** would include the rejected EC or issue accepted as EC. The accepted issue could be categorized as:

1) Needing development of counteractions,

2) Needing additional details for EC formalization,

3) Accepted issue which doesn't need counteraction development or additional specification but which is ready to progress to engineering implementation of change.

Select and develop counteraction

These processes are not included within the scope of the software solution. They are event based. Hence, their standardization with the software solution would be technically and economically prohibitive. The inputs to this process/activity include: 1)an accepted issues which needs interactions defined and 2)disapproved EC with recommendation for counteractions from specify, document , track and decision change process/activity. The recommendations are based on analysis by the EC team with the respect to cost, time, quality, and system effects. This mechanism/users include issue initiator and initiator's manager.

Specify document, track and decision change

Specifying document, tacking and decision change process/activity is initiated by an issue accepted as EC or recommended countermeasures. This process/activity is controlled by organizational procedures. The mechanism/users included issue initiator, EC team and action responsible. The outputs of this process/activity include an approved EC or a canceled EC. An approved EC is based on an evaluation from the EC team.

Engineering implementation of change

Engineering implementation of change is controlled by organizational procedures. The mechanisms/users include action responsible for engineering, supplier and customer. The outputs of this process/activity include a released EC, an implemented EC or a canceled EC. Release EC is entail issuance of work order from engineering to manufacturing. The triggers for this process/activity an issue accepted as EC or an approved EC.

Manufactory implementation of change

The mechanism/users include action responsible for manufacturing, supplier and customer. The outputs of this process/activity include a implemented EC or canceled EC. The trigger for this process/activity as EC or an approved EC.

ECMS Senior Design

Final Report CIS 4952/4962
December 17, 2012

Janel Gumbs
Abdoul Razakou Hamissou
Elisa Miller
Arth Suthar

Table of Contents

1.0 Introduction

Engineering change is defined as a modification of the product definition of a product due to quality enhancement, cost reductions, corrections of errors, supply resource, legal compliance, customer request, and process requests. The need of this in small the medium size business is crucial. Enterprise solutions are available but they are very expensive.

1.1 Goals and Objective

The goal of our software is to provide an open source or low cost engineering change management system for small to mid-size businesses; it is based on the Microsoft Dynamic NAV module[1].

1.2 Statement of Scope

The Engineering Change Management Software (ECMS) process is categorized into five following steps:

Identify need for engineering change

Identify the need for engineering change will be trigger by an issue(**input**), This process and activity is controlled by organizational procedure. The mechanisms/users include issue initiator, initiator's manager. An issue could be described as a product related concern with a potential of creating a change in the form, fit or function. The **output** would include the rejected EC or issue accepted as EC. The accepted issue could be categorized as:

1) Needing development of counteractions,
2) Needing additional details for EC formalization,
3) Accepted issue which doesn't need counteraction development or additional specification but which is ready to progress to engineering implementation of change.

Select and develop counteraction

These processes are not included within the scope of the software solution. They are event based. Hence, their standardization with the software solution would be technically and economically prohibitive. The inputs to this process/activity include: 1)an accepted issues which needs interactions defined and 2)disapproved EC with recommendation for counteractions from specify, document , track and decision change process/activity. The recommendations are based on analysis by the EC team with the respect to cost, time, quality, and system effects. This mechanism/users include issue initiator and initiator's manager.

Specify document, track and decision change

Specifying document, tacking and decision change process/activity is initiated by an issue accepted as EC or recommended countermeasures. This process/activity is controlled by organizational procedures. The mechanism/users included issue initiator, EC team and action responsible. The outputs of this process/activity include an approved EC or a canceled EC. An approved EC is based on an evaluation from the EC team.

Engineering implementation of change

Engineering implementation of change is controlled by organizational procedures. The mechanisms/users include action responsible for engineering, supplier and customer. The outputs of this process/activity include a released EC, an implemented EC or a canceled EC. Release EC is entail issuance of work order from engineering to manufacturing. The triggers for this process/activity an issue accepted as EC or an approved EC.

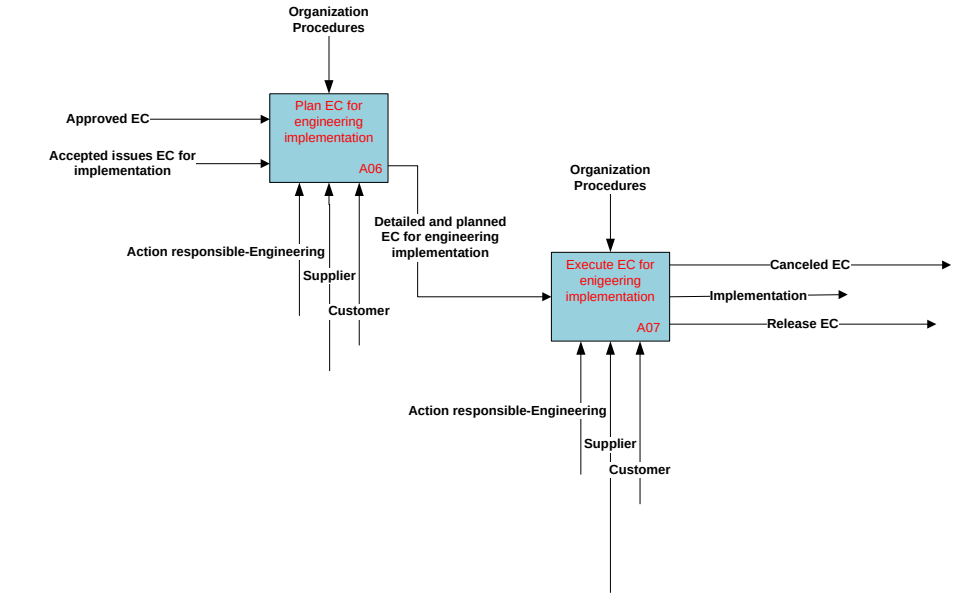
Manufactory implementation of change

The mechanism/users include action responsible for manufacturing, supplier and customer. The outputs of this process/activity include a implemented EC or canceled EC. The trigger for this process/activity as EC or an approved EC.

[1] Due to budget restrictions, the ECMS team decided against using Microsoft Dynamic NAV module.

Breakdown of Engineering implementation of Change (end of Software Solution)

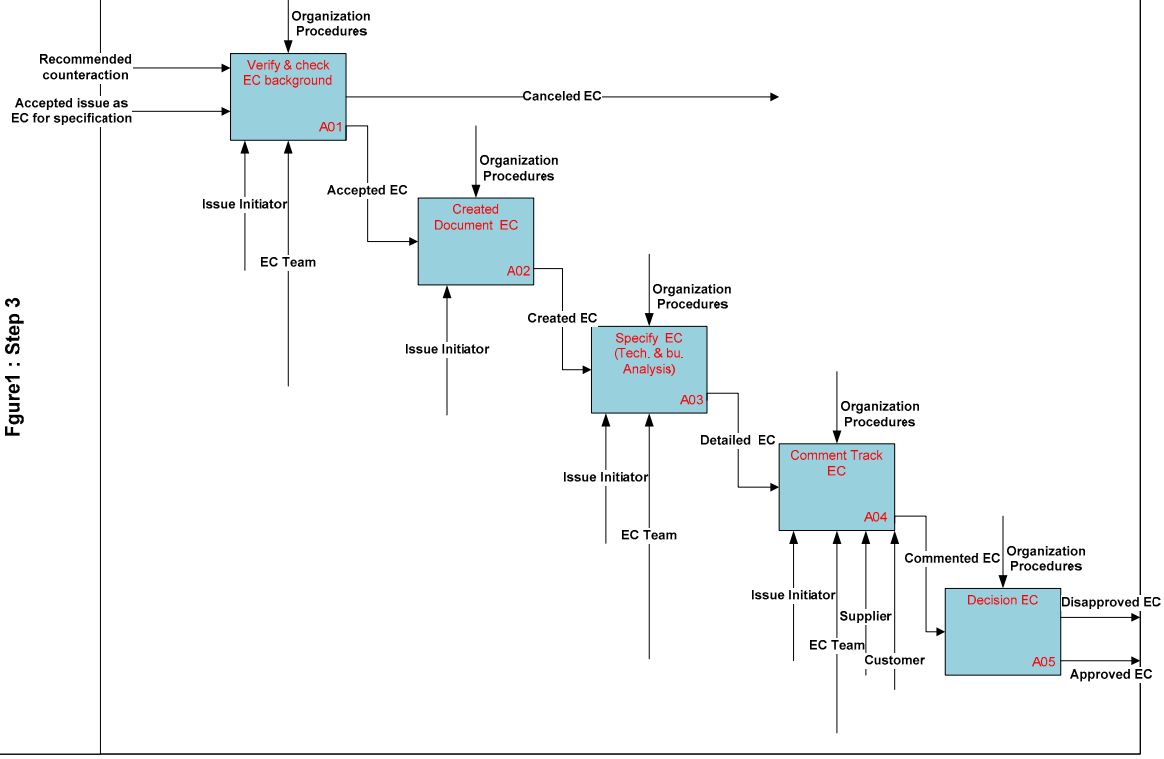
Fgure2 : Step 4



Note: Outputs of A07 are archived

Breakdown of Process /Activity: Specify, Document, Track and Decision change (beginning of Software Solution)

Fgure1 : Step 3



1.3 Software Context

This ECMS can be placed in various industries, ranging from engineering firms to medical practices. The software will reduce the time and money needed to verify background information, to plan implementation, to make decision and collaborate on engineering change tracking. This leads to more efficient handling of changes, happier customers and efficient use of company resources.

1.4 Major Constraints

Most engineering firms operate on windows machines, therefore the need of developing the software on windows platform is required.
The software solution architecture should be modeled as a three tier web based system
based on Microsoft's Windows distributed internet applications. Four components are included in
the software solution include: a database server, a web server, an application server, and a
number of clients/users.
The software must have a modular approach in order for the application to be industry non-specific this allows for the software to be applied in various scenarios.
Only authorized users can initiate, accept, reject, and cancel changes.
The database where the changes are kept must be backed up on a daily basis to keep the system updated and secure.

2.0 Usage Scenario

2.1 User profiles

Workers (Actors)

Department / Position	**General Impact of Project**
EC Team	Specifies, documents and decides upon an EC Request
Issue Initiator	Discovers an issue and initiate a change
Initiator's Manager	Approves acceptance of an issue of an EC
Action Responsible	Completes assigned engineering or manufacturing tasks

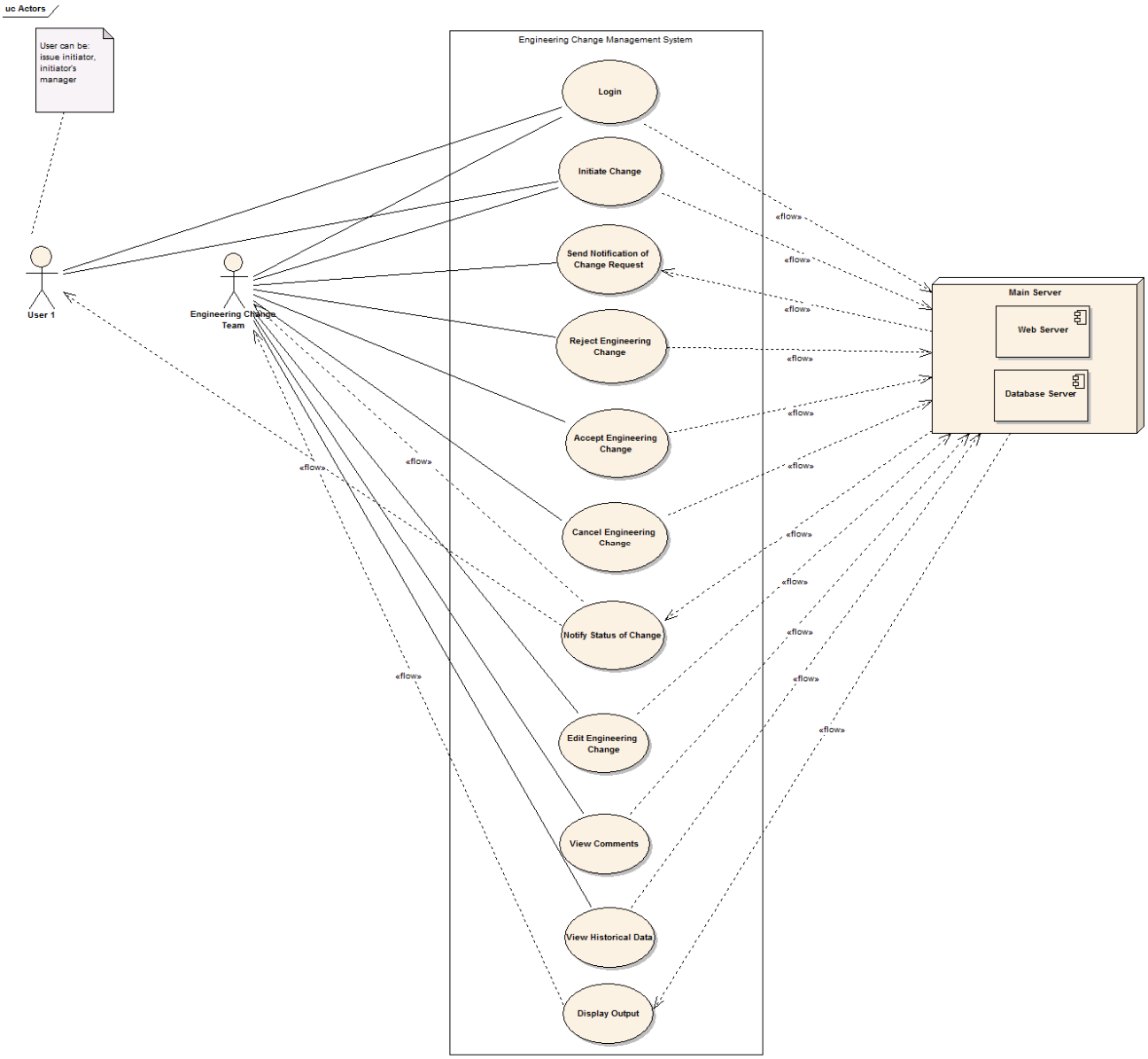
Business Actors

Actors	**General Impact of Project**
Customers	Accepts products or services in lieu of financial consideration
Suppliers	Provides products or services in lieu of financial consideration

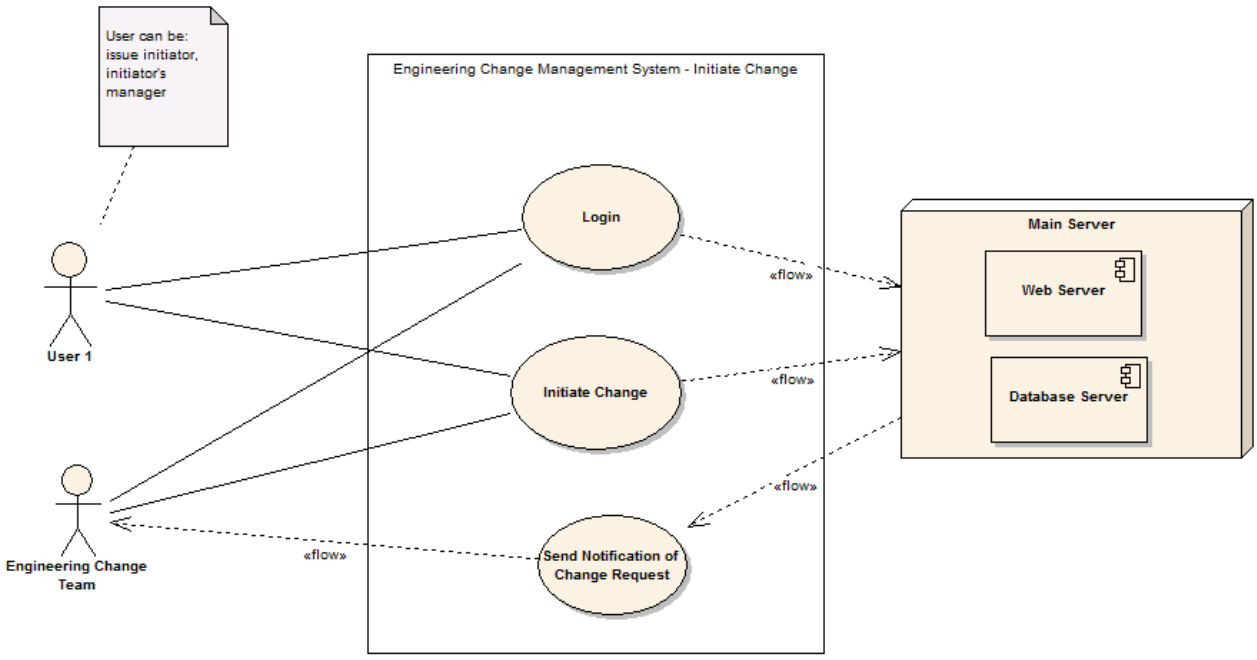
Other Systems

System	**General Impact of Project**
Product definition database within ERP System	Contains data necessary to precisely define a product for example drawings, test specs etc. Software solution needs to connect to this database to attach brief EC related information – EC id number, EC description to product part number via XML.

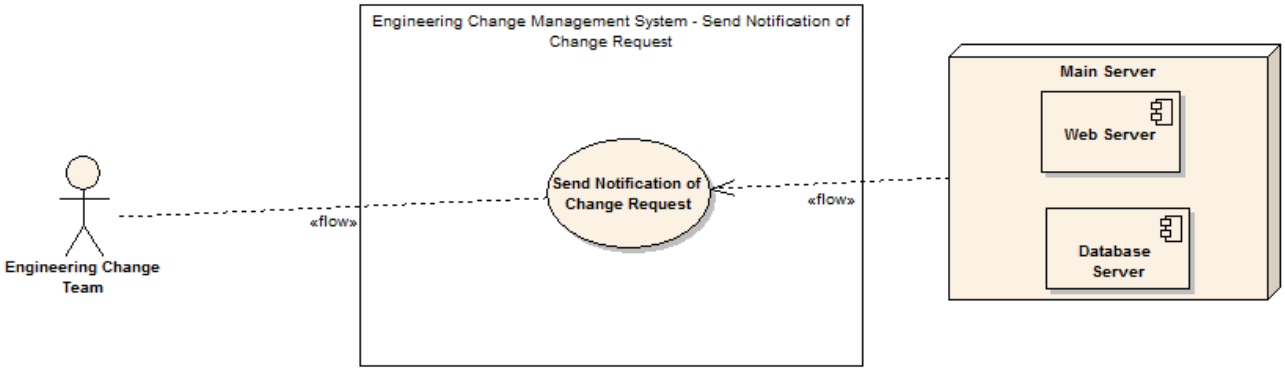
2.2 Use-Cases


uc Actors
User can be: issue initiator, initiator's manager
User 1
Engineering Change Team
Engineering Change Management System
Login
Initiate Change
Send Notification of Change Request
Reject Engineering Change
Accept Engineering Change
Cancel Engineering Change
Notify Status of Change
Edit Engineering Change
View Comments
View Historical Data
Display Output
Main Server
Web Server
Database Server
«flow»

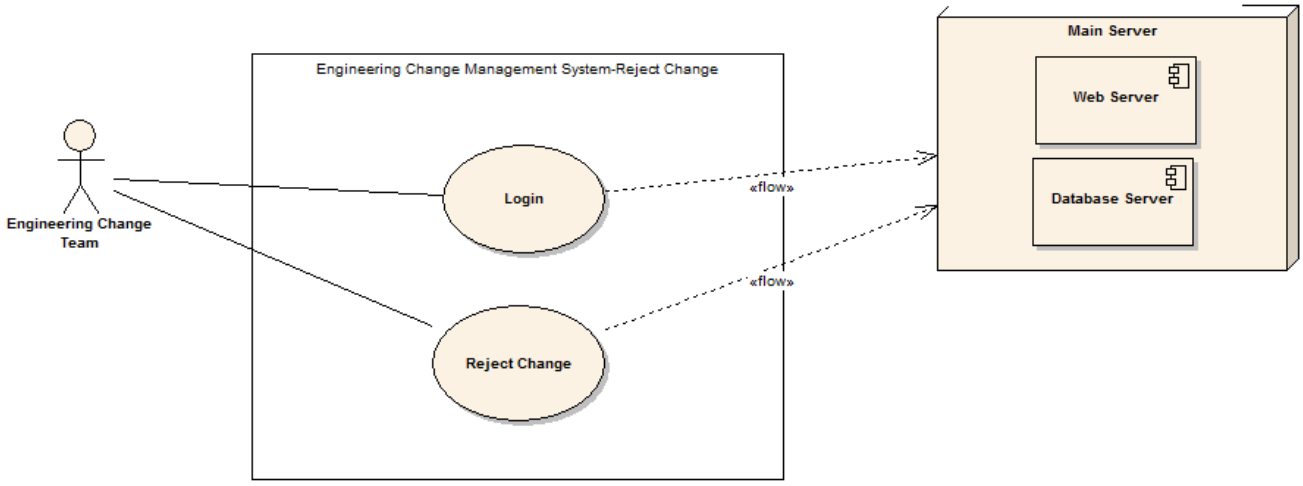
Use Case	Initiate Change
Description	Need for Engineering Change Identified
Actors	Engineering Change Team, Issue Initiator, Initiator's Manager , System (EC software)
Assumptions	1. The user has proper privilege levels to access the system and create document 2. Beginning of the Change Management Process-At this time, and engineering issue has been identified and needs to be put into the system for review
Steps	1. Login to the system 2. Select Create new Document 3. Enter Change Request Information 4. Save Changes 5. System Sends Notification to EC



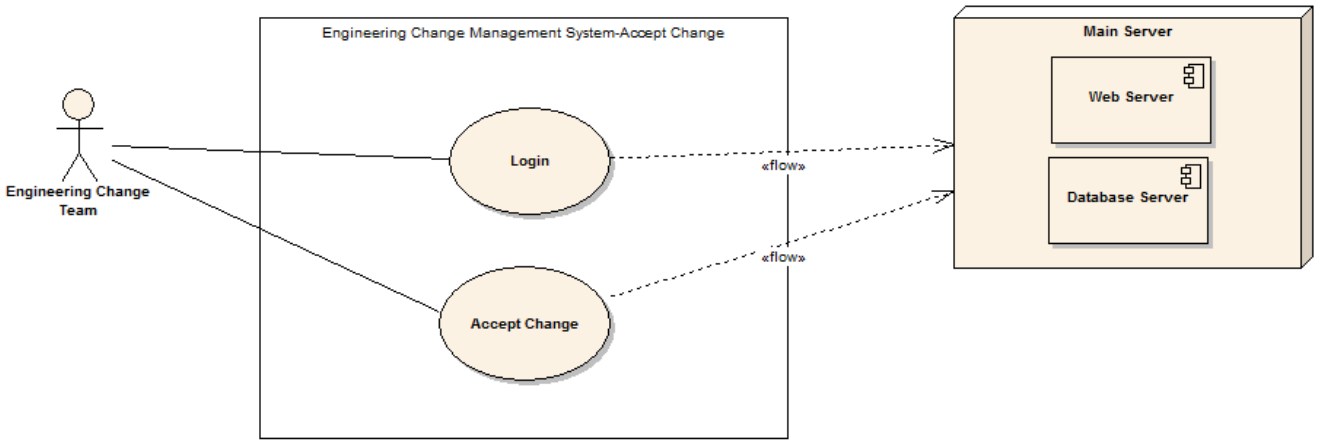
Use Case	Send Notification of Change Request
Description	Change has been initiated in the system, document created. (waiting on Reject or Accept Status from EC Team)
Actors	Engineering Change Team
Assumptions	1. The user has proper privilege levels to access the system and edit document 2. Change has been saved in the system
Steps	1. Once a new document has been created and saved, the system retrieves a list of all parties that need to be notified of change 2. System sends Message to all parties who can approve or reject change 3. Display confirmation that the message has been sent



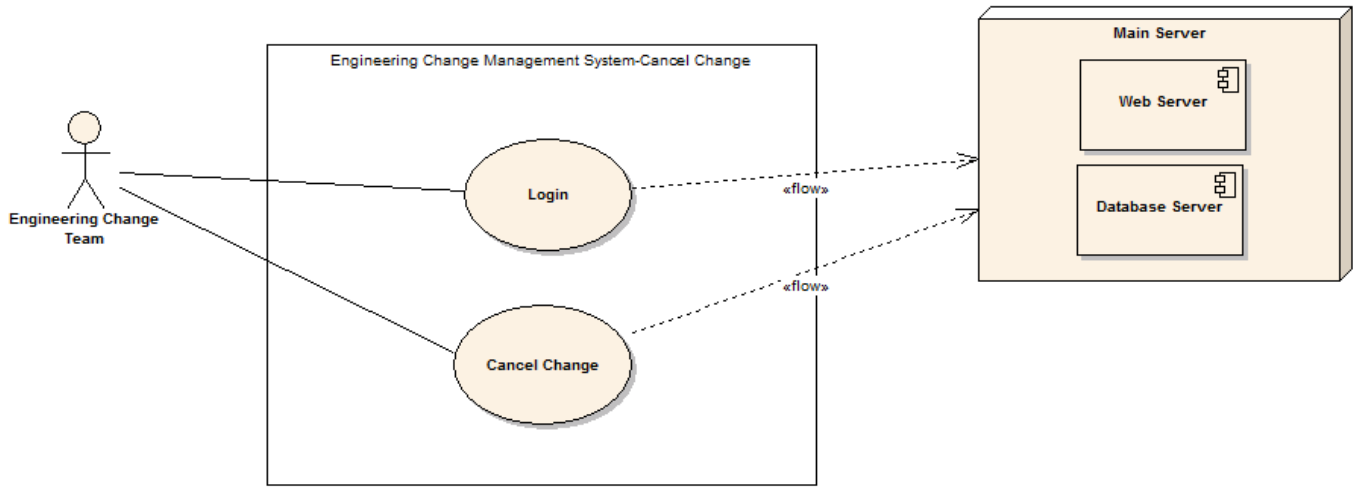
Use Case	Reject Engineering Change
Description	Message has been sent to EC Team stating that a new engineering change is waiting for review. EC Team reviews change, and determines that it cannot be implemented by the Engineers
Actors	Engineering Change Team , EC system
Assumptions	1. System sends to proper parties 2. Outside Review of Engineering Change Request has taken place 3. User has proper admin privileges to reject change within the system
Steps	1. Login to System 2. Select Engineering Change to be Rejected 3. Reject the change 4. Insert Reason for rejection 5. Update the system to reflect the rejection of Engineering Change



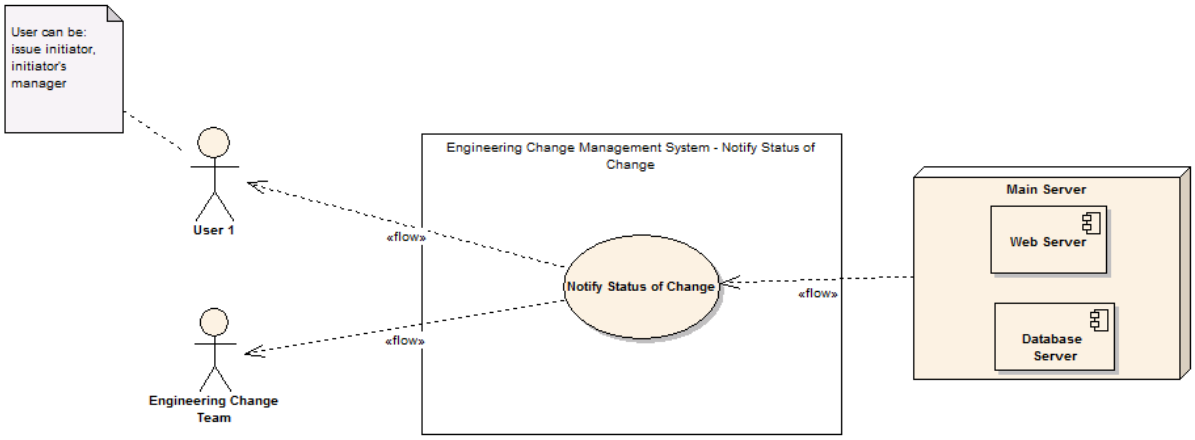
Use Case	Accept Engineering Change
Description	Message has been sent to EC Team stating that a new engineering change is waiting for review. EC Team reviews change, and determines that it can be implemented by the Engineers
Actors	Engineering Change Team
Assumptions	1. Notification has been sent to proper parties 2. Outside Review of Engineering Change Request has taken place 3. User has proper admin privileges to accept change within the system
Steps	1. Login to System 2. Select Engineering Change to be Accepted 3. Accept the change 4. Update the system to reflect the Acceptance of the Engineering Change Request



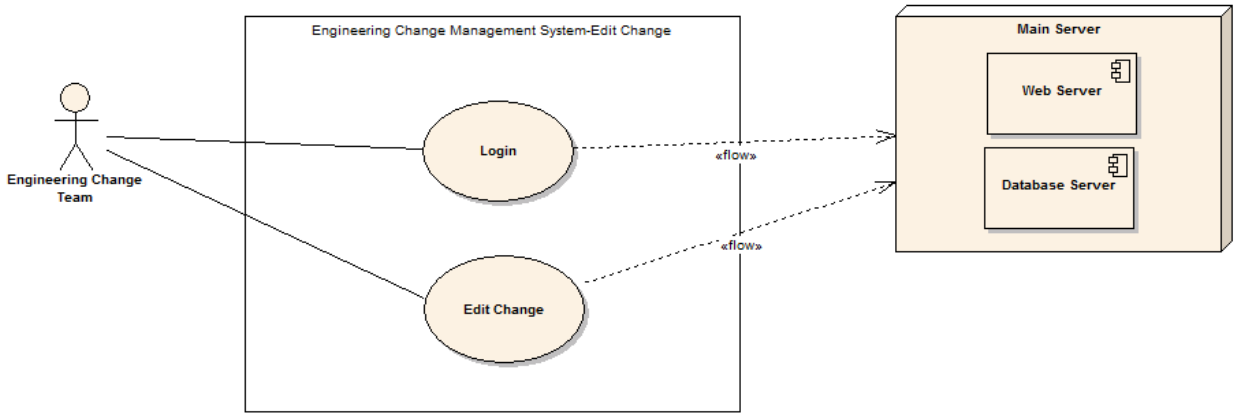
Use Case	Cancel Engineering Change
Description	EC Team has reviewed change, and determined that it can be implemented by the Engineers, more information has been received making it necessary to cancel the Engineering Change
Actors	Engineering Change Team
Assumptions	1. Notification has been sent to proper parties and Engineering Change is being implemented by the engineering teams 2. Outside Review of Engineering Change Request has taken place 3. User has proper admin privileges to accept change within the system
Steps	1. Login to System 2. Select Engineering Change to be Canceled 3. Cancel the change 4. Insert Reason for cancellation 5. Update the system to reflect the Cancellation of the Engineering Change Request



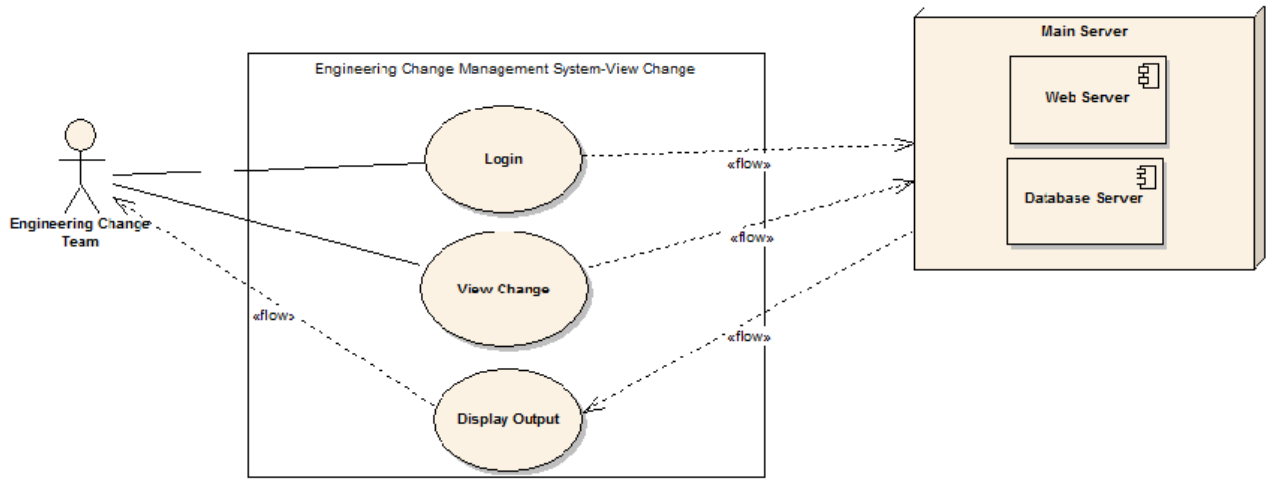
Use Case	Notify Status of Change
Description	Engineering Change Request has been filed in the system. The system will now notify the Engineering Change team or other initiator of the new status
Actors	Engineering Change Team, Issue Initiator, Initiator's Manager
Assumptions	1. Outside Review of Engineering Change Request has taken place 2. User has proper admin privileges to view changes within the system
Steps	1. The system retrieves a list of all parties that need to be notified of change 2. Message is sent to all parties who are involved with the initial request 3. Display confirmation that message has been sent



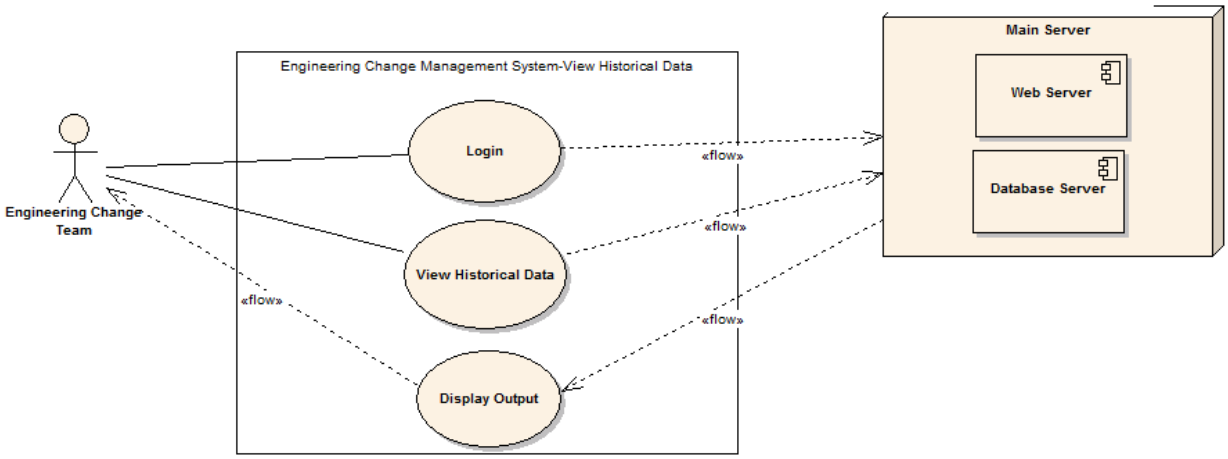
Use Case	Edit Engineering Change
Description	The EC Team places a note/comment on a specific document
Actors	Engineering Change Team
Assumptions	1. Outside Review of Engineering Change Request has taken place 2. User has proper admin privileges to edit documents within the system
Steps	1. Login to System 2. Select Engineering Change to place comment 3. Add Comment 4. Save comment in the system



Use Case	View Comments
Description	The EC Team has placed a note/comment on a specific document-User request to view comment
Actors	Engineering Change Team
Assumptions	1. The EC Team has placed a note/comment on a specific document 2. User has proper admin privileges to view comments within the system
Steps	1. Login to System 2. Select Engineering Change to view comment 3. View Comment



Use Case	View Historical Data
Description	The EC Team request to view all actions made on the Engineering Change thus far
Actors	Engineering Change Team
Assumptions	1. The user has proper privilege levels to access the system and view documents 2. Engineering Change has been initiated and saved in the system
Steps	1. Login to System 2. Select Engineering Change to view historical data 3. View data



2.3 Special Usage Considerations

Users with basic Windows and application software experience should be able to operate system.
User must know how to use Microsoft Dynamic NAV[2]
Files generated should be scannable with Trend Micro OfficeScan Client
Application must run on Windows XP or higher, system

3.0 Data Model and Description

3.1 Data objects and descriptions

IssueInitiator Object
Department – Name of the department of person initiating change
FirstName – First name of the person initiating change
LastName – Last name of the person initiating change
LoginName – A unique login for the person initiating change
Password – Password for logging into system
Title – Title of the person initiating change

InitiatorManager Object
FirstName – First name of the manager
LastName – Last name of manager
LoginName – A unique login for the manager
Password – Password for logging into system
Title – Title of the manager

Supplier Object
CompanyName – Name of the company of the supplier
FirstName – First name of the person from supplier
LastName – Last name of the person from supplier
LoginName – A unique login for the supplier
Password – Password for logging into system
Speciality – Type of speciality of the supplier

Customer Object
CompanyName – Name of the company of the customer
FirstName – First name of the customer
LastName – Last name of the customer
LoginName – A unique login for the customer
Password – Password for logging into system
Speciality – Type of speciality of the customer

EngChange Object
CancelDate – Date the engineering change was cancelled
CreateDate – Date of when the engineering change was created
EngChangeID – A unique ID of the engineering change submitted
EngChangeTitle – Name of the engineering change
ImplementDate – Date of when the engineering change was implemented
InitiatorFirstName - First name of the person that initiated change

[2] Due to budget restrictions, the ECMS team decided against using Microsoft Dynamic NAV module.

InitiatorLastName - Last name of the person that initiated change
InitiatorMgrFirstName – First name of initiator's manager
InitiatorMgrLastName – Last name of initiator's manager
RejectDate – Date of when the engineering change was rejected

EngChangeTeam Object
Department – Name of the department of engineering change team
FirstName – First name of the person from the engineering change team
LastName – Last name of the person from the engineering change team
LoginName – A unique login for the person from the engineering change team
Password – Password for logging into the system
Title – Title of the person from the engineering change team

ActionResponsible Object
FirstName – First name of initiator
LastName – Last name of initiator
LoginName – A unique login to log into the system
Password – Password for logging into the system
Title –Title of person of initiator

ResponsibleSpecify Object
FirstName – First name of initiator
LastName – Last name of initiator
LoginName – A unique login to log into the system
Password – Password for logging into the system
Title –Title of person of initiator

ResponsibleEng Object
FirstName – First name of person on Engineering team
LastName – Last name of person on Engineering team
LoginName – A unique login to log into the system
Password – Password for logging into the system
Title –Title of person on Engineerng team

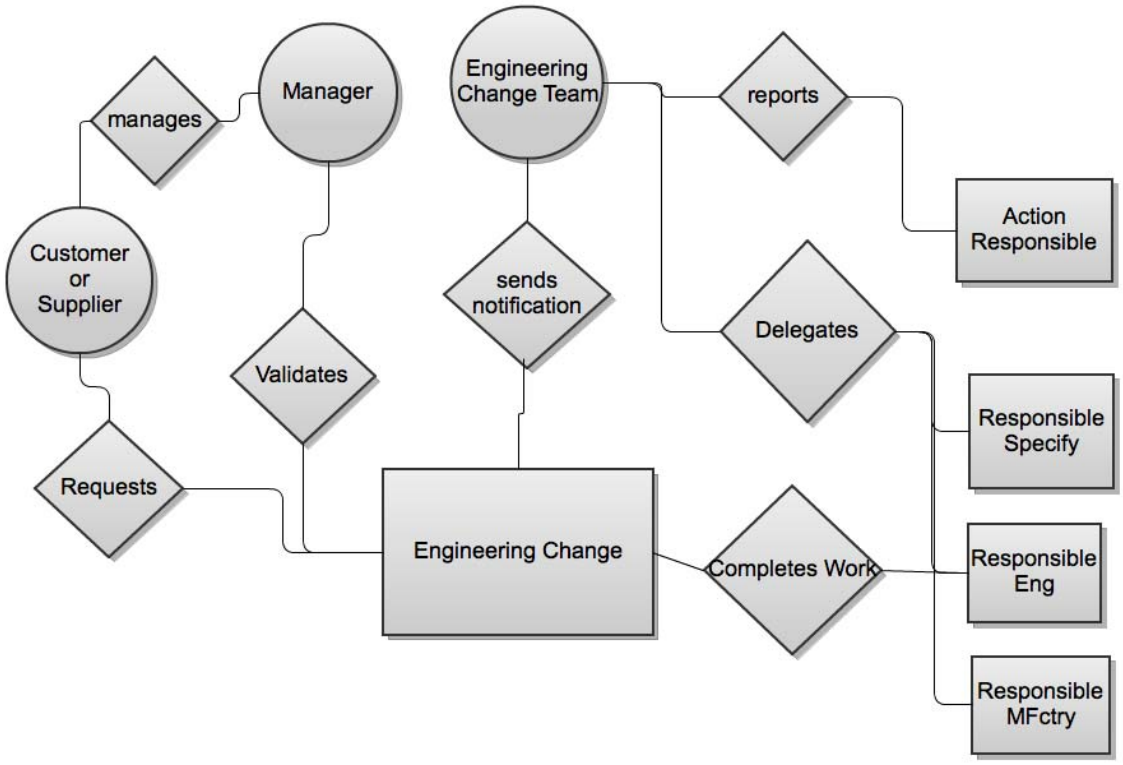
ResponsibleMFactory Object
FirstName – First name of person on manufacturing team
LastName – Last name of person on manufacturing team
LoginName – A unique login to log into the system
Password – Password for logging into the system
Title –Title of person on manufacturing team

3.1.1 Relationships

An **IssueInitiator** initiates an issue and can be a **Customer** or **Supplier**, and is managed by the **IntiatorManager**. If the **InitiatorManager** approves the issue, it is identified as an **EngChange** (Engineering Change). The **EngChange** can not exist without an **IssueInitiator** initiating an issue (or submitting a request for **EngChange)** and if it is not validated by the **InitiatorManager.**

Notification is sent to the **EngChangeTeam** (Engingeering Change Team) that an **EngChange** has been submitted. **ActionResponsible** reports the Engineering Team. **ResponsibleSpecify** and **ResponsibleEng** completes EC work and reports to **EngChangeTeam** before validation. **ResponsibleMFactory** also reports to the **EngChangeTeam.**

3.1.2 Complete Data Model



3.1.3 Data Dictionary

Data that will be stored:

- Description of Engineering Change
- Status of EC
- Comments
- Historical data
- Implementation/approval/rejection/cancelled dates
- Keep track of submitter, approver, EC
- Information about submitter, approver, department

4.0 Functional Model and Description

4.1 Description for Function Initiate Change ()

A user logs in to the system to start the process for engineering change. The user can be any of the following actors: Engineering Change Team, Issue Initiator, and Initiator's Manager.

4.1.1 Processing narrative (PSPEC) for Initiate Change

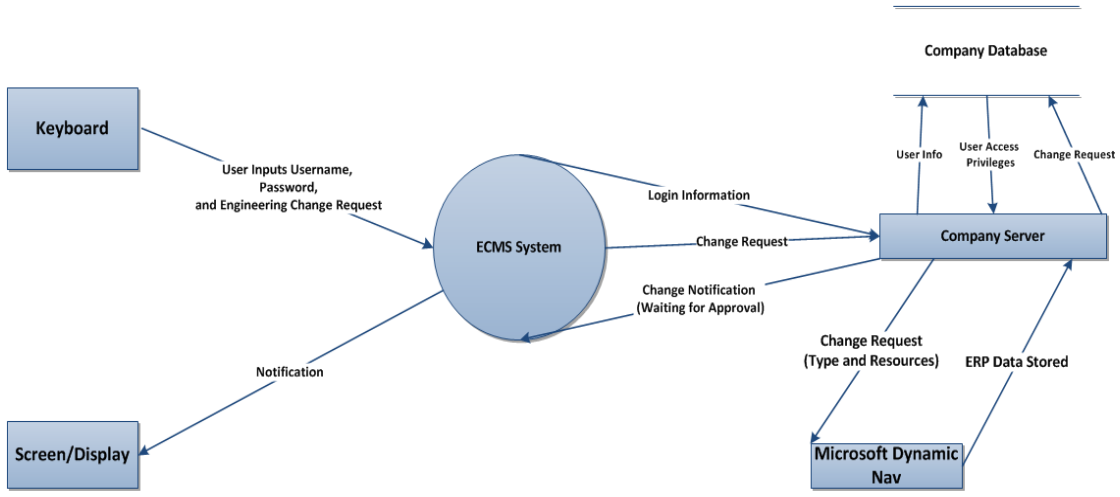
The user logins to the system and enters user name and password (the number of characters/ digits required will depend on password complexity needed to meet the company's security standards). The information is then verified via database and checked (and crossed checked) for two different purposes:

1. Authorization to enter the system
2. Admin Level (will vary depending on the user's predetermined access to the system)

After this is verified, the user can choose to create a new document. In this document, the user enters change request information and saves the file to the company's database server. (ERP data is prepared and stored for review)

Once the request is saved, a notification is sent (from the company server) to the EC Team Members who are authorized to approve/reject/cancel the ECMS request.

4.1.2 Initiate Change flow diagram



4.1.3 Initiate Change interface description

The inputs to this function are username, password and engineering change request. Outputs are the notification to specific EC team members and Change request data store to company server.

4.1.4 Initiate Change transforms

There are no transforms for the Initiate change function.

4.1.5 Performance Issues

Response times for notifications should be within a reasonable timeframe (under 3 seconds for login verification and under 10s for Change request submission notification)

4.1.6 Design Constraints

Unauthorized access to the system will be prevented through log in verification. Poor network connectivity will also disrupt system usage.

4.2 Description for Function View Change()

Any authorized user can log on to view and manage engineering changes.

4.2.1 Processing narrative (PSPEC) for function View Change()

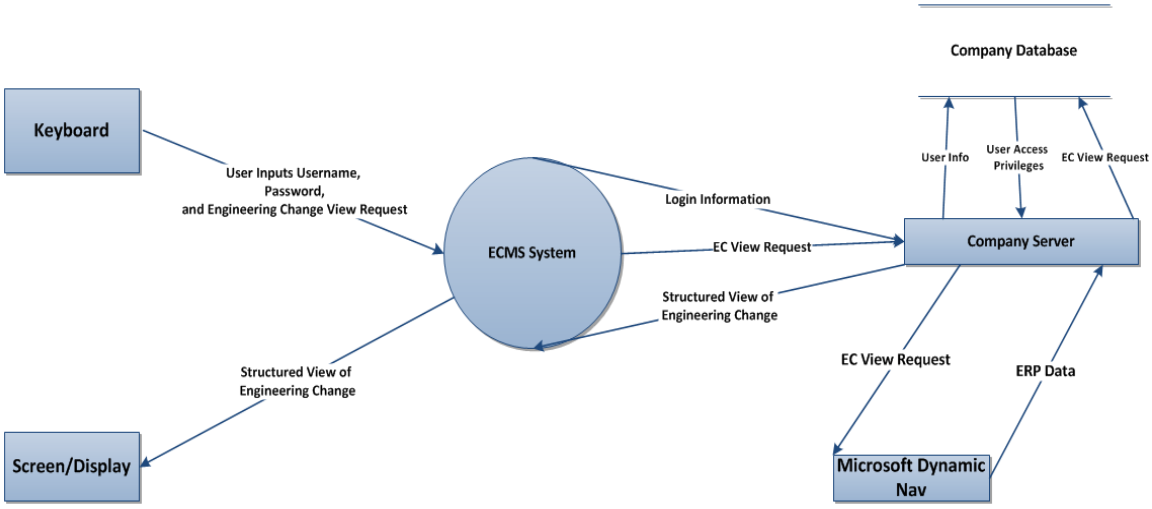
A processing narrative for function View Change() is presented:

The user logins to the system and enters user name and password (the number of characters/ digits required will depend on password complexity needed to meet the company's security standards). The information is then verified via database and checked (and crossed checked) for two different purposes:

1. Authorization to enter the system
2. Admin Level (will vary depending on the user's predetermined access to the system)

After this is verified, the user can select an engineering change file to view. If the user has proper admin level he/she can view the engineering changes and files associated with it.

4.2.2 Function View Change() flow diagram



4.2.3 Function View Change() interface description

The inputs for the View Change () function are user name, user password, and the information needed to view a specific engineering change. The outputs are a structured view of the engineering change.

4.2.4 Function View Change() transforms

The transforms (sub-functions) within the view change function are as follows:

1. Approve
2. Reject
3. Cancel
4. Edit
5. Generate report

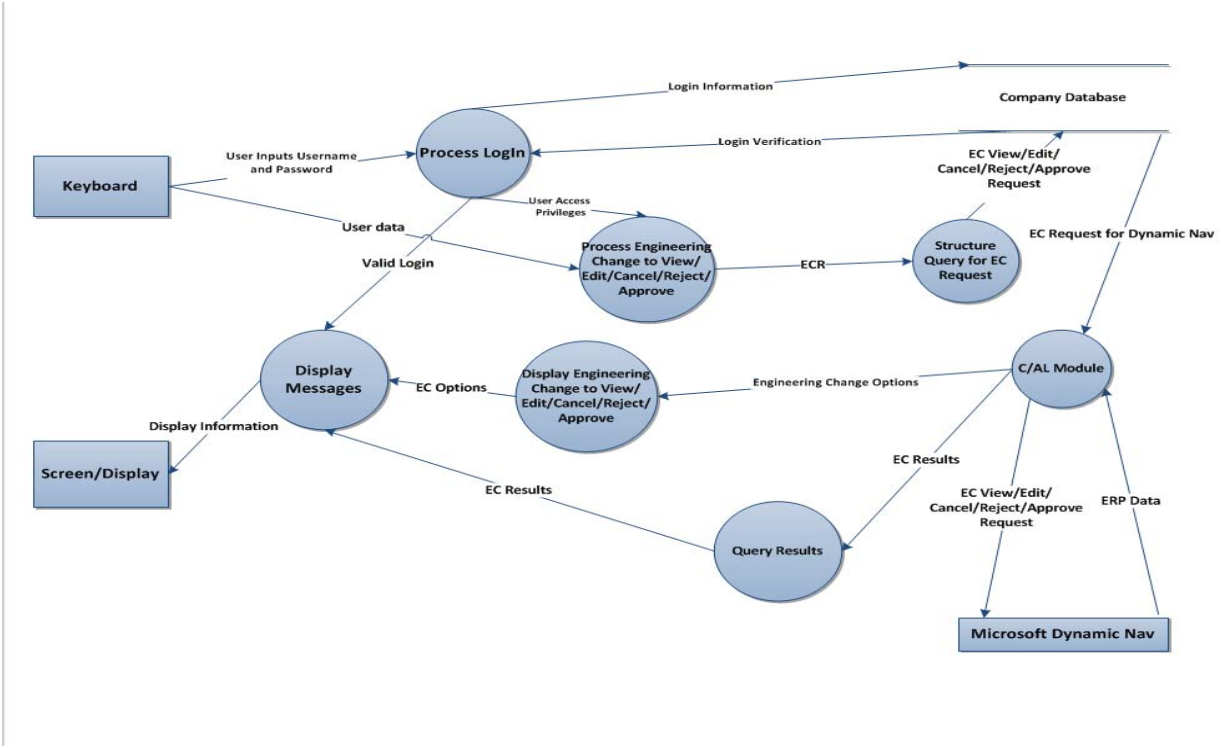
4.2.4.1 Transform Approve() description (processing narrative, PSPEC)

The user will log in to the ECMS System and select the corresponding change to be approved. Once selected, the information will be sent to the system for a status update and data stores are updated. Notification is sent to the user to verify approval.

4.2.4.2 Transform Approve() interface description

The inputs for the Approve () function are user name, user password, and the information needed to view a specific engineering change. The outputs are notification of approval.

4.2.4.3 Transform Approve() lower level flow diagrams



4.2.4.4 Transform Reject() description (processing narrative, PSPEC)

The user will log in to the ECMS System and select the corresponding change to be rejected. Once selected, the information will be sent to the system for a status update and data stores are updated. Notification is sent to the user to verify rejection of engineering change.

4.2.4.5 Transform Reject() interface description

The inputs for the Reject () function are user name, user password, and the information needed to view a specific engineering change. The outputs are notification of engineering change rejection.

4.2.4.6 Transform Reject() lower level flow diagrams

See flow diagram for approve()

4.2.4.7 Transform Cancel() description (processing narrative, PSPEC)

The user will log in to the ECMS System and select the corresponding change to be canceled. Once selected, the information will be sent to the system for a status update and data stores are updated. Notification is sent to the user to verify cancellation.

4.2.4.8 Transform Cancel() interface description

The inputs for the Cancel () function are user name, user password, and the information needed to view a specific engineering change. The outputs are notification of engineering cancellation.

4.2.4.9 Transform Cancel() lower level flow diagrams

See flow diagram for approve()

4.2.4.10 Transform Edit() description (processing narrative, PSPEC)

The user will log in to the ECMS System and select the corresponding change to be edited. Once selected, the information will be sent to the system for a status update and data stores are updated. Notification is sent to the user to verify that an edit has been made.

4.2.4.11 Transform Edit() interface description

The inputs for the Edit () function are user name, user password, and the information needed to view a specific engineering change. The outputs are notification of engineering change edits.

4.2.4.12 Transform Edit() lower level flow diagrams

See flow diagram for approve()

4.2.4.13 Transform Generate Report() description (processing narrative, PSPEC)

The user will log in to the ECMS System and select the corresponding changes to be included in the report. Once selected, the information will be sent to the system for a status update and data stores are updated. Report is generated and information is sent back to the user per request.

4.2.4.14 Transform Generate Report() interface description

The inputs for the Cancel () function are user name, user password, and the information needed to view a specific engineering change(s). The outputs are an engineering change report.

4.2.4.15 Transform Generate Report() lower level flow diagrams

4.2.5 Performance Issues

Per client requirements:

”The single/specific engineering change report generation should not take more than 3 seconds of processing time. A report containing 100 engineering changes should not take more than 8 seconds of processing time. A report containing more than 100 engineering changes should not take more than 60 seconds of processing time. Searching a single keyword should not take more than 3 seconds of processing time. A complex search should not take more than 60 seconds of processing time.” (2012 Preetinder Gill, PhD Student at Eastern Michigan University)

Response times for notifications should be within a reasonable timeframe (under 3 seconds for login verification and under 10s for Edit/Approve/Reject/Cancel notification)

4.2.6 Design Constraints

Unauthorized access to the system will be prevented through log in verification. Poor network connectivity will also disrupt system usage.

For organizing database objects, C/AL Programming must be used for coding modules within Microsoft Dynamic Nav[3].

4.3 Software Interface Description

4.3.1 External machine interfaces

No other external machine interfaces will be needed for the ECMS system.

4.3.2 External system interfaces

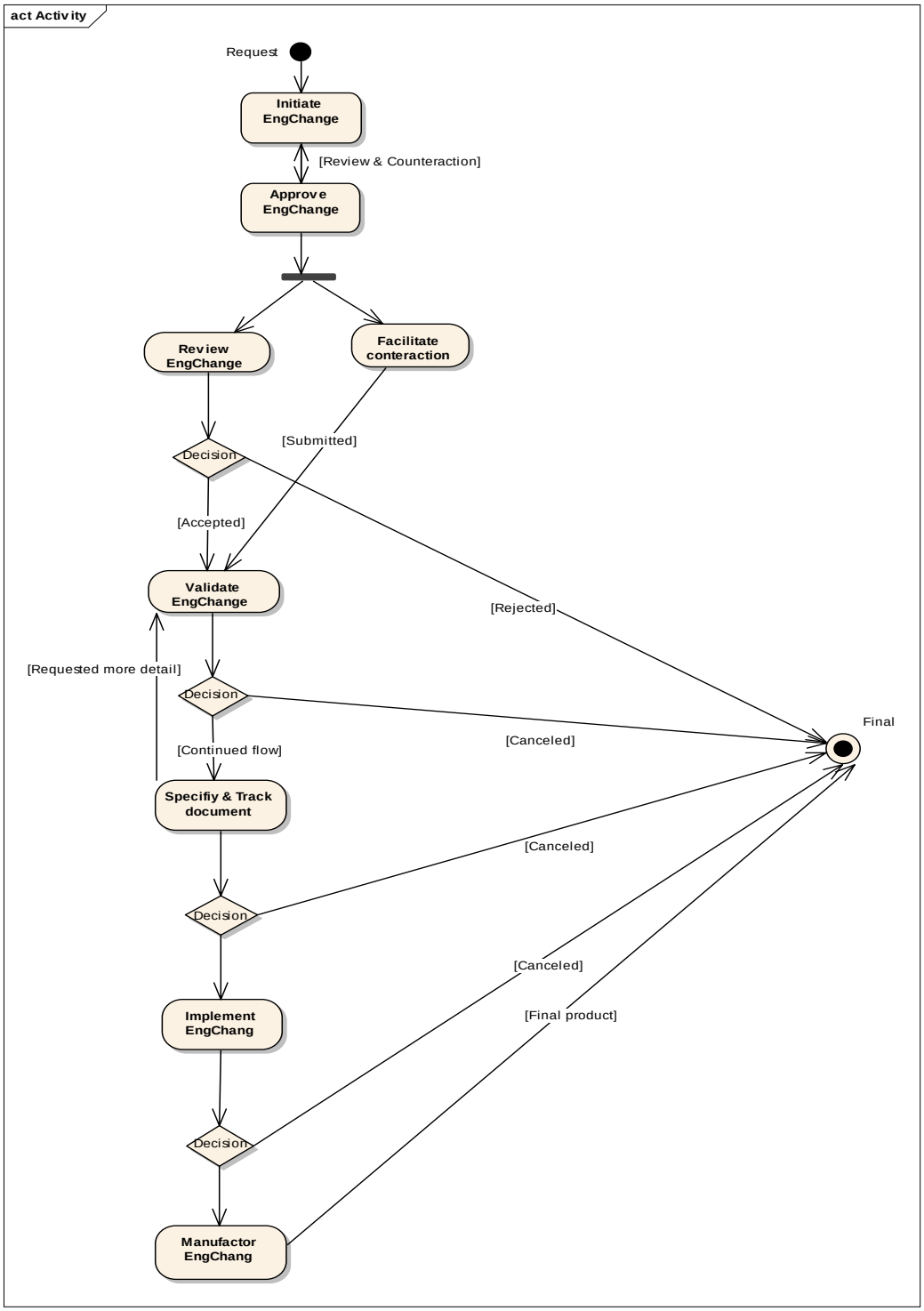
Using client server architecture, the client will connect to company server and dynamic nav modules through the internet using TCP/IP protocols.

4.3.3 Human interface

All human interfaces will are built as web interfaces using html and server side scripting language.

[3] Due to budget restrictions, the ECMS team decided against using Microsoft Dynamic NAV module.

4.4 Control flow description



5.0 Behavioral Model and Description

The software process proceeds as follows: it starts from the issue initiation, then the validation by the initiator Manager takes place, this is followed by the EC team approval, next it moves to the Engineering Team software component for implementation validation, and finally to the Manufactory team interface where the manufacturing of “EngChange” will be decided.

5.1 Description for software behavior

The Major events include:

- ”EngChange” initiation
- Approval by the initiator manager
- Check for any counteraction or additional information
- “EngChange” review Engineering Change Team and validate
- Completion of the EC work assign to Action Responsible for document specification and tracking
- “EngChange” received by the Engineering Team and completion of the work assigned to Action Responsible Engineering
- After the validation the “EngChnage” moves to the Manufactory Team where the “EngChange” will finally be manufactory or canceled, after the completion of Action Responsible Manufactory .

5.1.1 Events

There are 6 mains events:

1. “EngChange” initiation
2. Initiator Manager approval
3. Checking for any counteraction or additional information
4. Engineering Change Team validation
5. Manufactory Team validation
6. EngChange Team decision

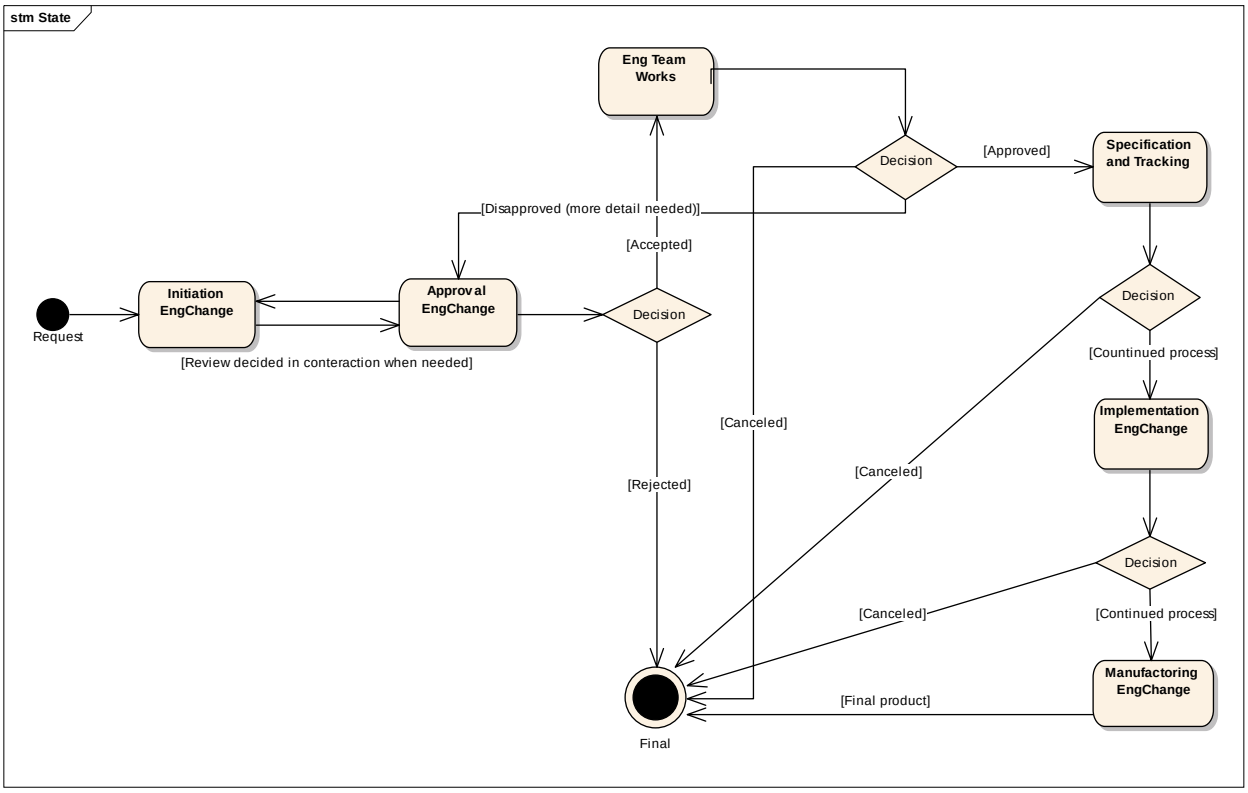
5.1.2 States

List of the State from State Diagram:

- Request for “EngChange”
- Issue initiation
- Approval “EngChange”
- Implementation or cancel of “EngChange”
- Manufacturing or cancel of “EngChange”
- Final (“EngChange” save in archive)

5.2 State Transition Diagrams

UML State diagram



5.3 Control specification (CSPEC)

- Every time a “EngChang” is initiated, the software sends notification to the Initiator Manager automatically
- When a decision is made about an “EngChang”, all the parties involved are notified
- After Manufactory Team validation of “EngChange”, the “Engchange” is saved in the archive

6.0 Restrictions, Limitations, and Constraints

There are several primary constraints to our system. Due to the timeline and size of our project, our final product is a prototype rather than a finished product. However, the prototype will be fully functioning with some minor features (that are not important to functionality) left out. We are also limited to a small budget. We must be able to complete the project with limited grant money and whatever our personal budgets allow for. This limits our ability to develop the project to exact specifications. In addition, we are restricted to only tools that are compatible with Microsoft Dynamic NAV, with Dynamic NAV being a major requirement to the software any language, testing tool, or data used must conform to NAV[4]. Information of all EC's must be stored in a database and are only accessible by the software. Some constraints on the system will be the hardware on which the system is running. The speed of the system will most likely be determined by this factor. The final constraint is based on the how fast the network is; which determines the speed of the system.

7.0 Validation Criteria

7.1 Classes of tests

Unit testing on:

1. Submitting an EC
2. Receiving notification
3. Viewing EC
4. Editing EC
5. Logging into system
6. Accepting EC
7. Rejecting EC
8. Cancelling EC
9. Making comments

7.2 Expected software response

- Incorrect username/incorrect password
- Comments fields should be populated
  - If field is empty, message should appear to put in comment
- Reject/cancelled with comments
  - Comment/reason field should be populated, or rejection will not be submitted
  - EC submission should send out notification
    - Output that submission has been sent
    - EC team receives notification
  - Status of change should send out notification
  - Viewing historical data, invalid search produces no results, message should appear

[4] Due to budget restrictions, the ECMS team decided against using Microsoft Dynamic NAV module.

7.3 Performance bounds

The following performance requirements are based on the document provide by the client:

- The single/specific engineering change report generation should not take more than 3 seconds of processing time.
- A report containing 100 engineering changes should not take more than 8 seconds of processing time.
- A report containing more than 100 engineering changes should not take more than 60 seconds of processing time.
- Searching a single keyword should not take more than 3 seconds of processing time.
- A complex search should not take more than 60 seconds of processing time.
- Time needed to move data from one location to another will excluded from calculation of time needed to meet various performance requirements.
- The software solution is expected to support one transaction per second with 99% availability.

8.0 System traceability matrix

Our specification is not being developed for a product; therefore we have no Product Strategy. In addition, being a potential open source project, the cost to the user will be minimal. **See design section for a filled out table.**

ID	*Functional Requirement*	*Status*	*Technical Specification*	*Software Module(s)*	Tested In	Verification	Additional Comments
001	Initiate Change						
002	View Change						
003	Approve Change						
004	Reject Change						
005	Cancel Change						
006	Edit Change						
007	Generate Report						
008	Search Change						
009	Track Change						
010	Notification of Change						
011	Validate Change						
012	Login						

8.3 Analysis metrics to be used

8.3.1 Number of Transactions between Database and Module

This will allow us to know how many times the module communicates with our database. We have to make sure that these transactions are not only syntactically correct but also the correct information will pass through.

8.3.2 Number of Total EC Changes

This will allow us to know how many changes are made on a single EC, allowing us to determine how well the changes are being tracked, notified, verified, approved, or rejected.

8.3.3 Generating Overall Report

This will allow us to determine the total accuracy and efficiency of the software. By generating a report all the changes are listed and are able to match if they were done manually.

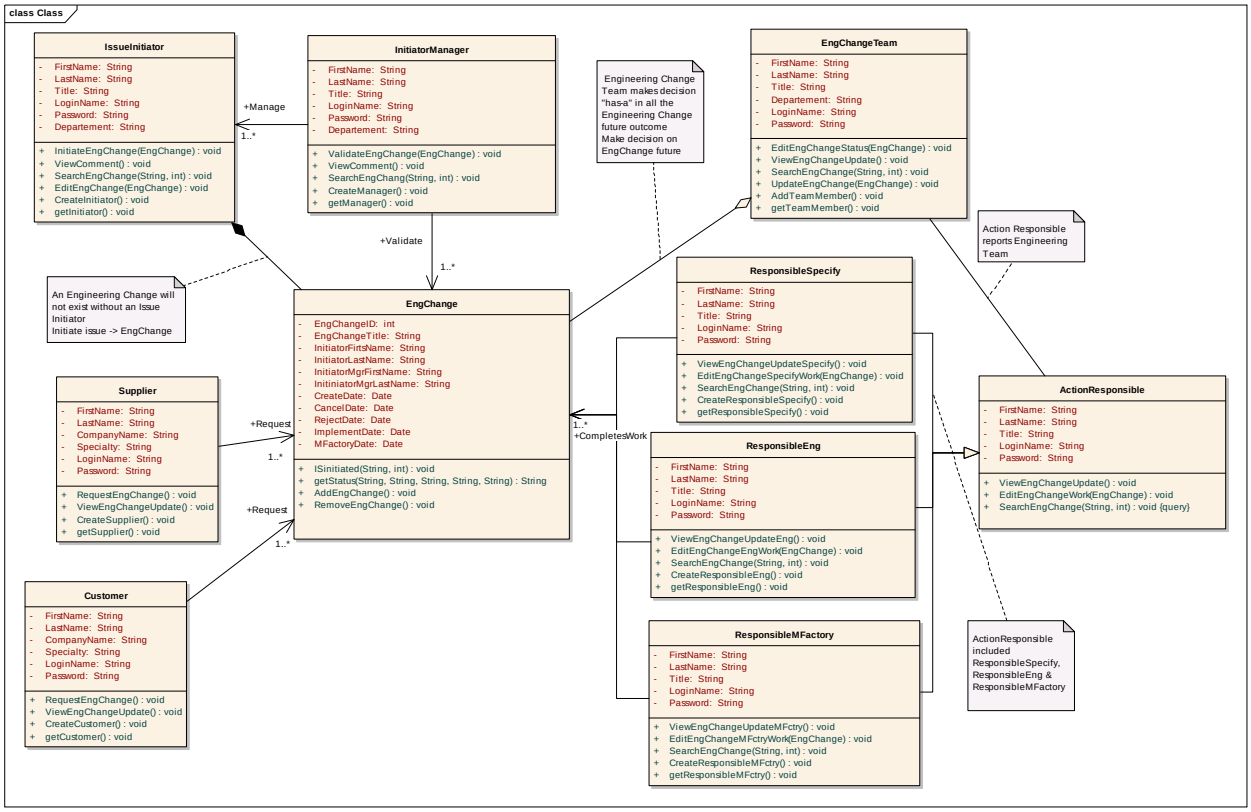
8.3.4 Number of Searches

This will allow us to determine the speed of our software; we must have our software perform searches within certain times based on the complexity of the search.

8.4 Supplementary information (as required)

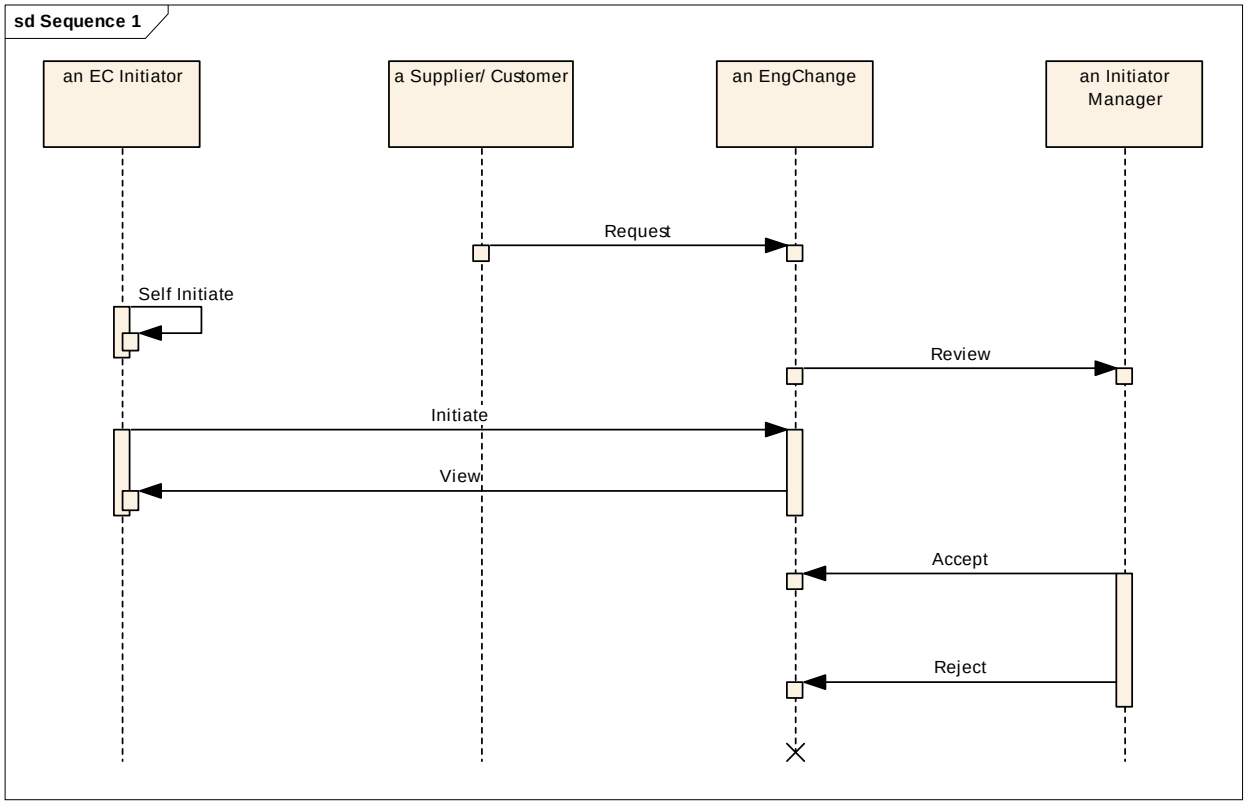
9.0 Requirements-UML Diagrams

9.1 Class Diagram

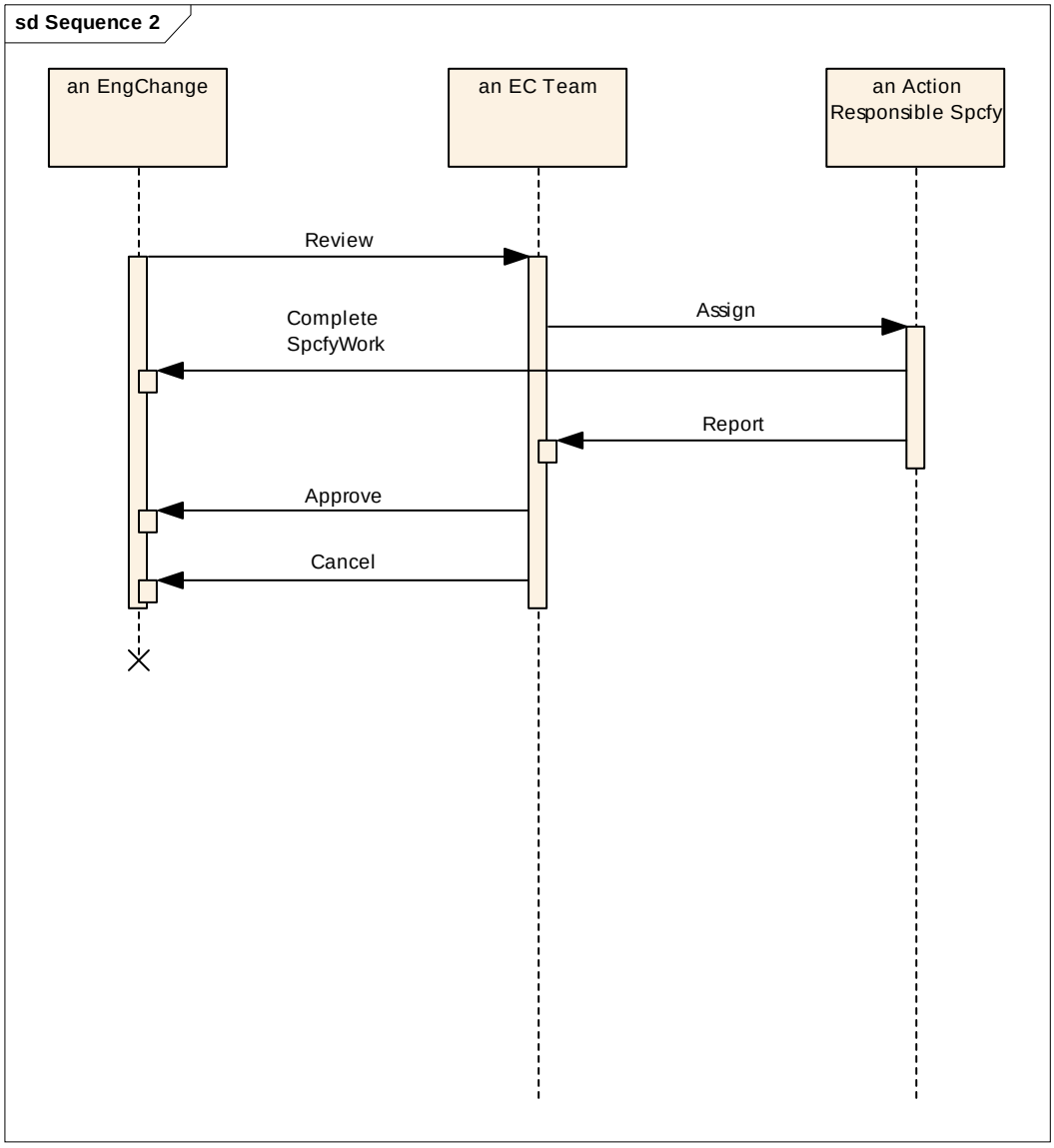


9.2 Sequence Diagrams

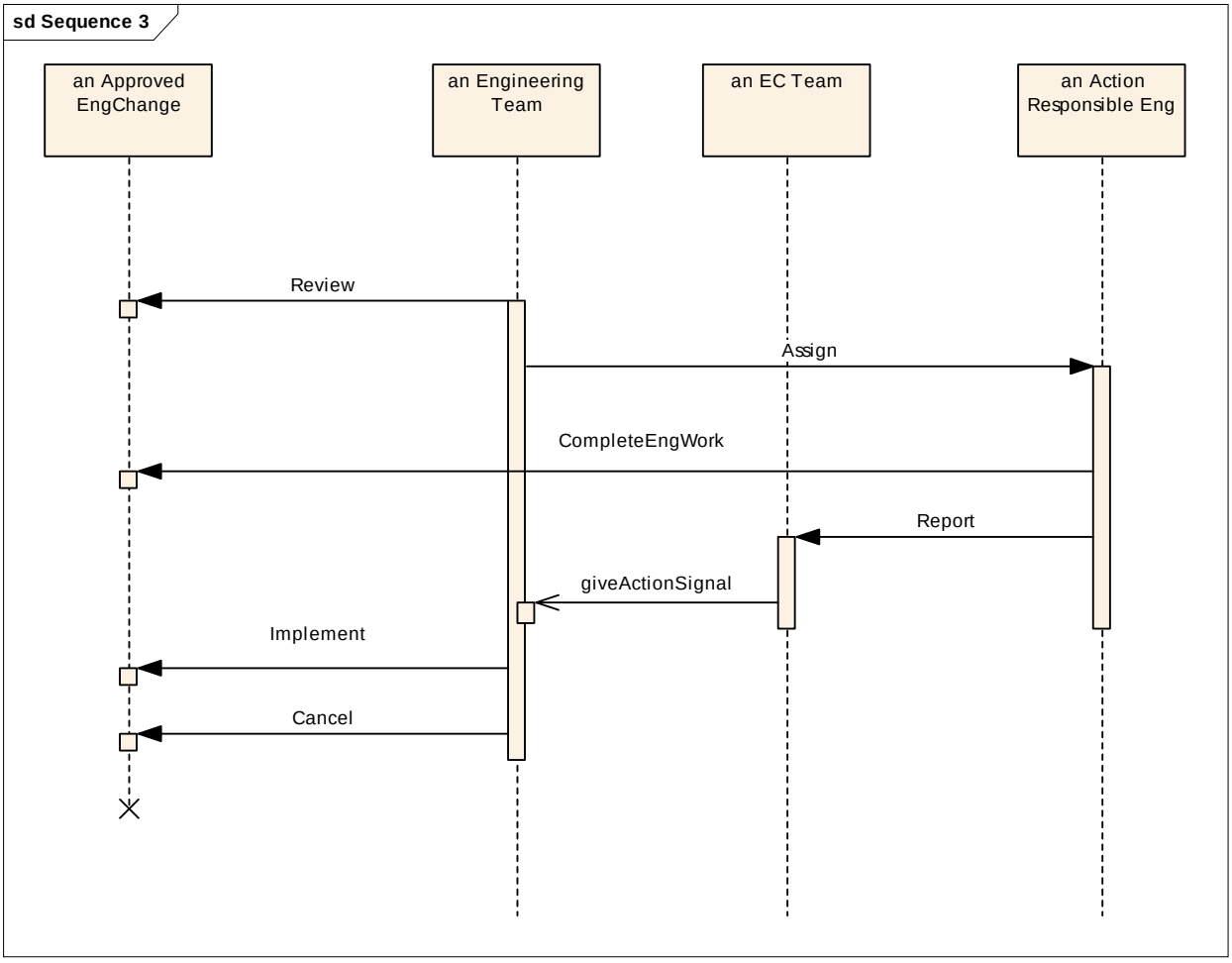
9.2.1 Sequence EngChange Initiation – Acceptance



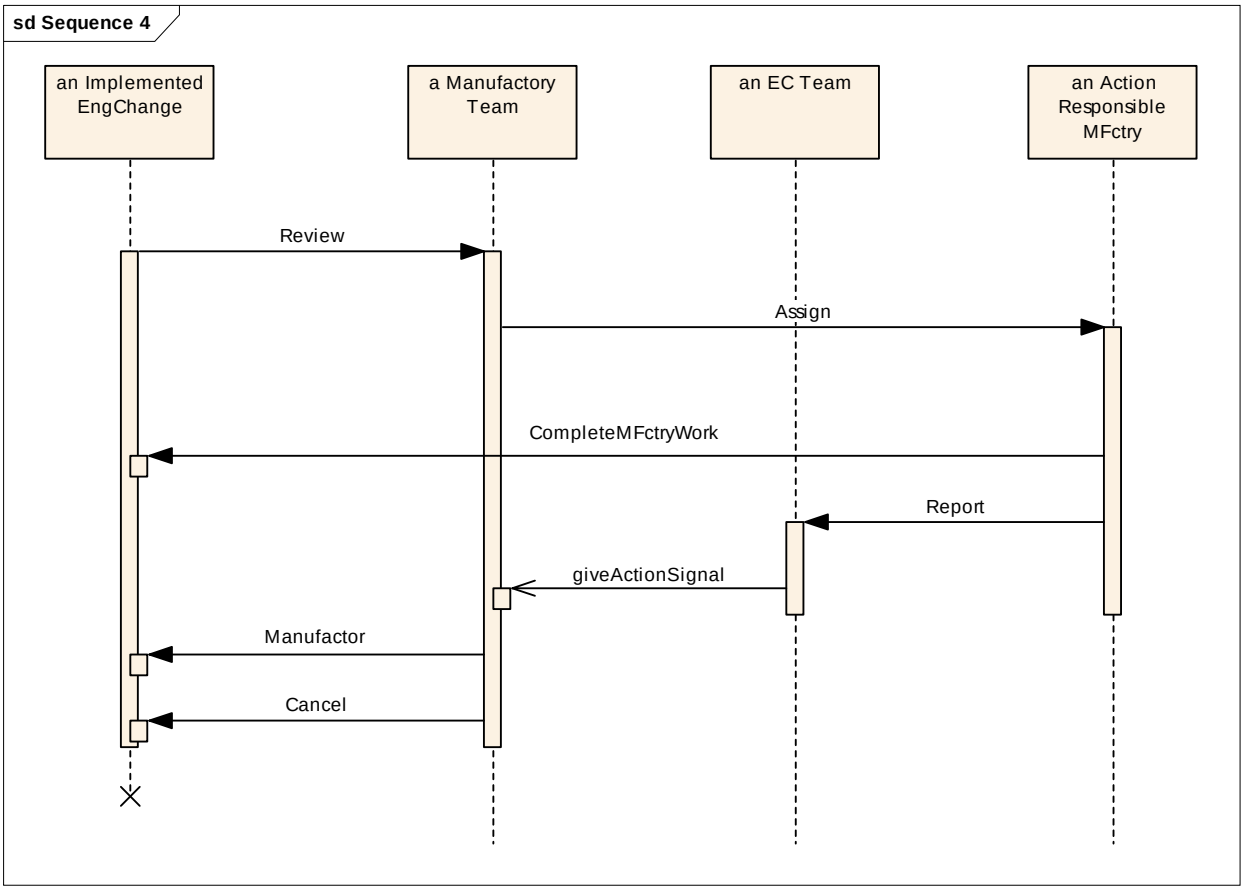
9.2.2 EngChange Approval - Document Specification



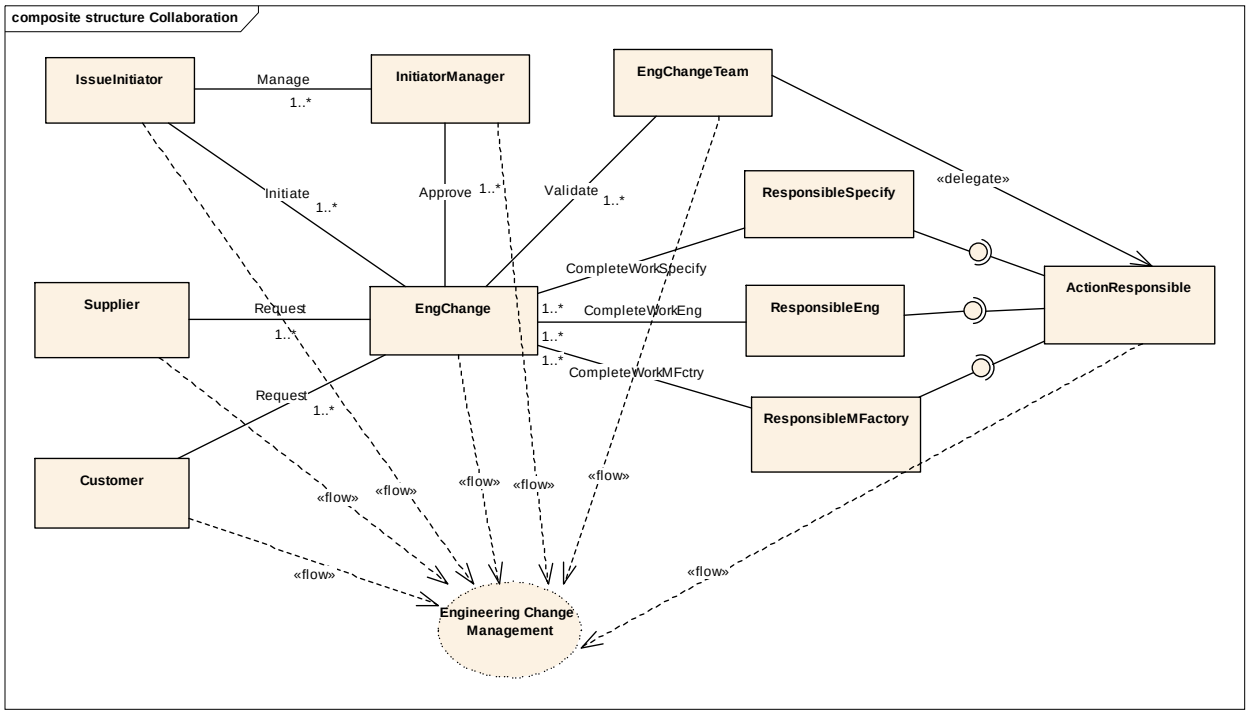
9.2.3 EngChange Approval – Implementation



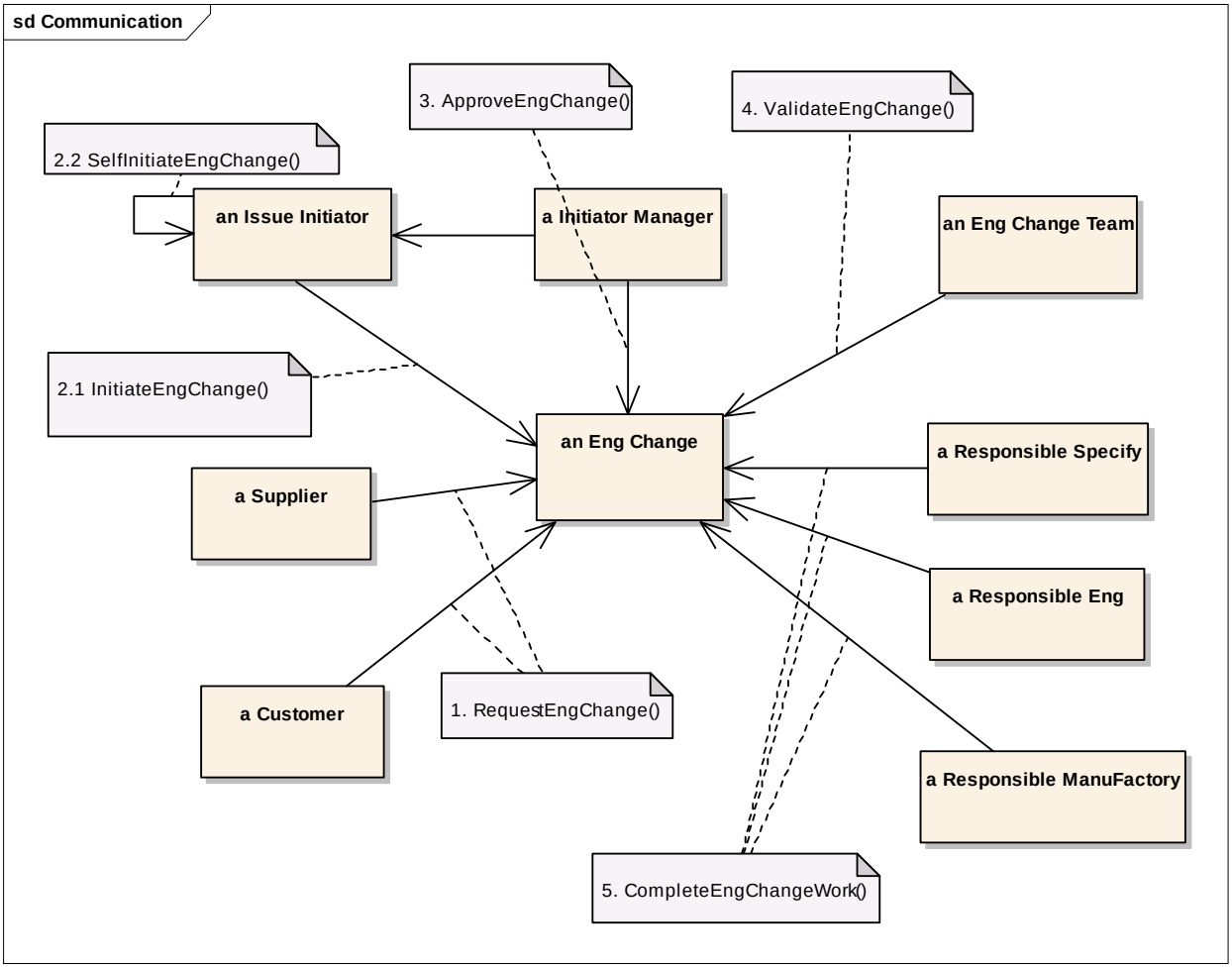
9.2.4 EngChange Approval – Manufactory



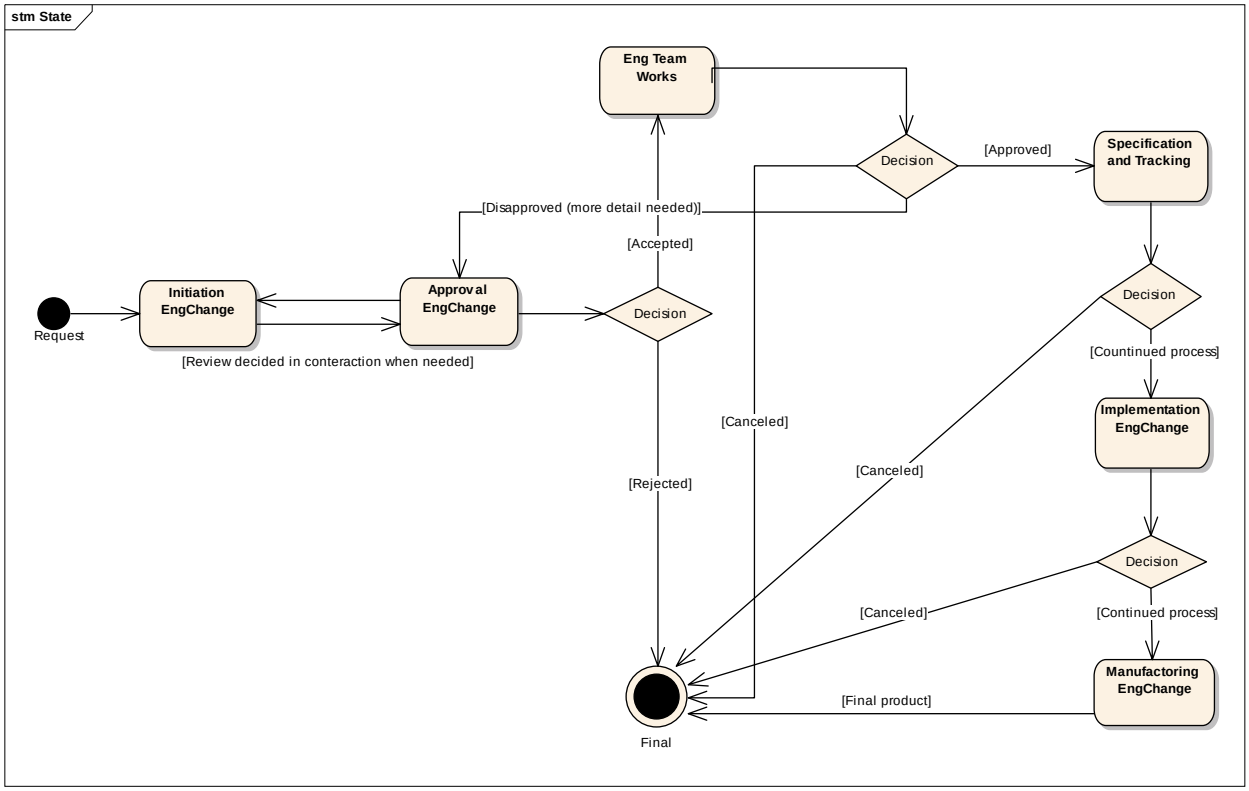
9.3 Collaboration Diagram



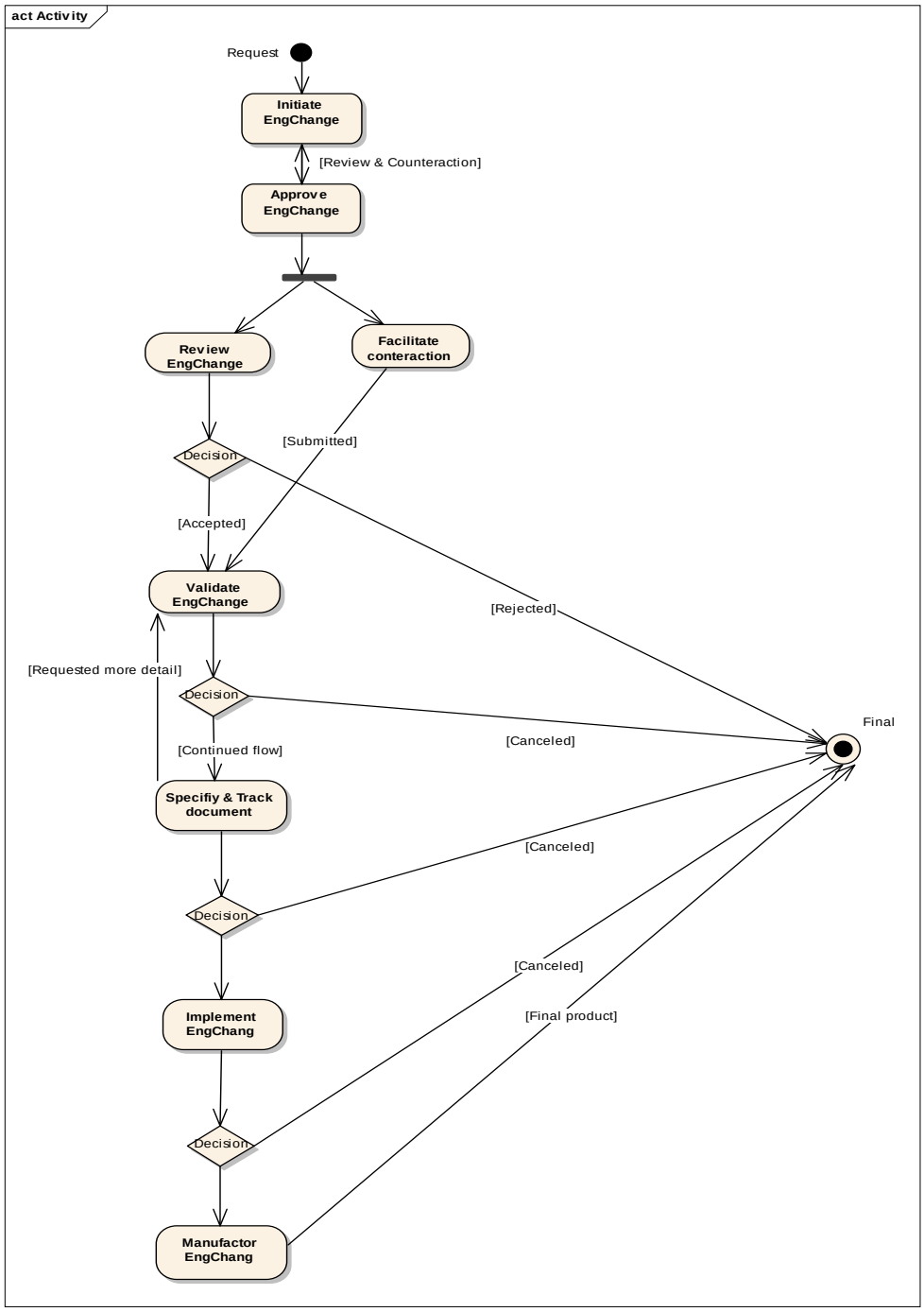
9.4 Communication Diagram



9.5 State Diagram



9.6 Activity Diagram



10.0 Software Project Plan Introduction

The project scope is composed of 5 categorized sequential processes: "**identification of engineering change"** process, which is trigger by the issue. The output of this process will either constitute the input of the "**selection and develop counteraction" process** if needed. Alternatively, it will trigger the next process, which is "**document specify, track and decision change**". Either of the aforementioned output processes can trigger the "**engineering implementation change"** process – its output can either be the cancellation of EC or "**manufactory implementation of change"**. The output of this process can either be implemented or canceled.

10.1 Project scope

The Engineering Change Management Software (ECMS) process is categorized into the five following steps. The output of the previous process will constitute an input to the next process or the following one .It should also be noted that all the processes/activities are controlled by the organization's procedures.

1. **Identify need for engineering change**.

The mechanisms/users include: issue initiator, initiator's manager, customer(s) and supplier(s).

Input: the issue
Outputs: issue accepted as EC.
The accepted issues could be categorized as:

- needing development of counteractions,
- needing additional details for EC formalization
- accepted issue which doesn't need counteraction development or additional specification, but which is ready to progress to engineering implementation of change

2. Select and develop counteraction
The mechanisms/users include: issue initiator and initiator's manager.

Inputs: an accepted issue (that needs counteractions defined), and disapproved EC with recommendations for counteractions from the- **specify, document, track and decision change** process/activity.
Outputs: the recommendations counteraction, which are based on analysis by the EC team with respects to cost, time, quality, and system effects.

3. Specify document, track and decision change
The mechanisms/users include: issue initiator, EC team and action responsible.

Input: issue accepted as EC or recommended countermeasures
Outputs: an approved EC or a canceled EC. An approved EC is based on an evaluation from the EC team.

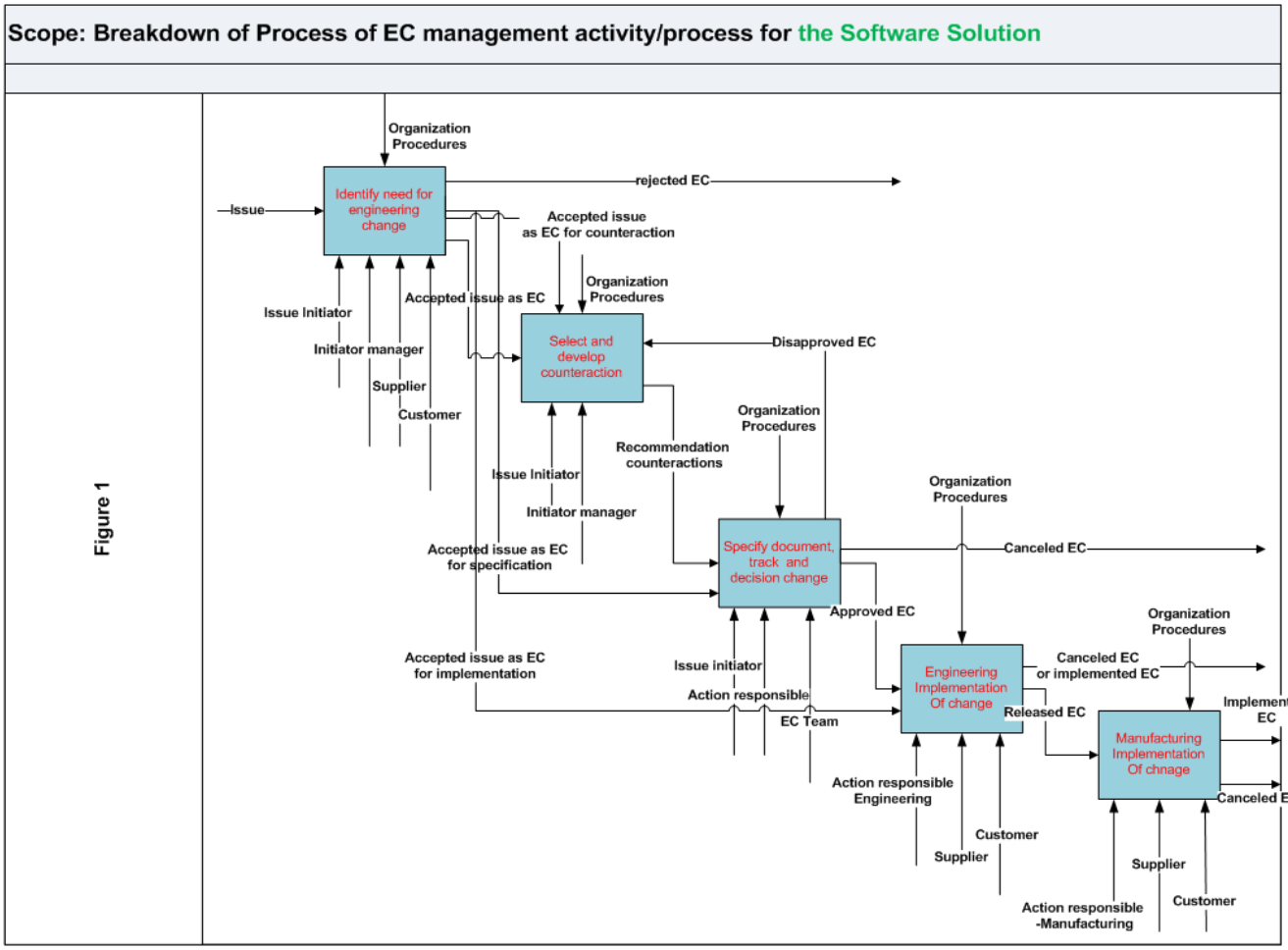
4. **Engineering implementation of change**
The mechanisms/users include: action responsible for engineering, supplier and customer.

Input: an issue accepted as EC or an approved EC.
Outputs: a released EC, an implemented EC or a canceled EC. Released EC entails issuance of a work order from engineering to manufacturing.

5. **Manufacturing implementation of change**
The mechanisms/users include: action responsible for manufacturing, supplier and customer.

Input: an issue accepted as EC or an approved EC.
Outputs: an implemented EC or a canceled EC.

ECMS Process Diagram:



10.2 Major software functions

The major functionalities of the Engineering Change software are:

1. Each staff member using the software system should be uniquely identified using proper login name and password.

2. The application provides a common interface that allows the user to log in all information related to engineering change. The information will include all past activities related to a specific EC previously entered in the system.

3. The system should allow the search and retrieval of the data when needed

4. The software shall generate automated specific EC report

5. The system shall generate notification to all the party involve automatically after the process is completed

6. The system shall allow an automated signature process when the EC moves from one team to the next (Ex. the process change moving from Engineering team to manufacturing team), if the EC is either rejected, approved or canceled.

7. The system should track all open and closed engineering change.

10.3 Performance/Behavior Issues

The following performance requirements are based on the document provide by the client:

- The single/specific engineering change report generation should not take more than 3 seconds of processing time.
- A report containing 100 engineering changes should not take more than 8 seconds of processing time.
- A report containing more than 100 engineering changes should not take more than 60 seconds of processing time.
- Searching a single keyword should not take more than 3 seconds of processing time.
- A complex search should not take more than 60 seconds of processing time.
- Time needed to move data from one location to another will excluded from calculation of time needed to meet various performance requirements.
- The software solution is expected to support one transaction per second with 99% availability.

10.4 Management and technical constraints

The EC software is a web-based application solution based on Microsoft Dynamic NAV, aimed to optimize engineering change process in order to improve the supply chain management in mid-sized organizations, the following are the major technical constraints:

- The software shall operate in all windows platform
- The software architecture solution should be modeled as a three tier layer system which included: a database server, a web service or technology providing web service and a database server.
- The software must have a modular approach, so its use is not limited to a specific industry
- Lack of availability of the Microsoft Dynamic NAV (SCM software) upon which the software should be built, may delay the project and add another major risk in the implementation.

- Lack of the data sample to elaborate comprehensible use cases.
- The users should have predefined disk quotas. However, the software solution needs to back up all data
- The cloud - based data backup is provided as SaaS by Carbonite.
- The system shall support at least 50 user accounts, along with 5 administrative accounts and 1master account
- The memory requirements to the application should not exceed 4 GB.
- Sub components of the software need to be modular based on service oriented architectures. Standard interfaces, such as XML, and protocols such as TCP/IP should be used to connect and link the software solution internally and externally.

11.0 Project Estimates

11.1 Historical data used for estimates

Historical data was difficult to find for this particular type of software, however a software with similar requirements but not similar output. Comparable systems include: Repositories, Version Control Systems, Work Flow Software, and Project Planning Tools. Most of these software do not have more than 4000 LOC and because ECMS is, in sorts, an aggregate of these software we do not expect the software to be any more than 6000 LOC.

11.2 Estimation techniques applied and results

FP-Based Estimation and Process Based estimation were used to analyze cost, effort and time estimates for the project. FP-Bases Estimates assess inputs, outputs, inquiries, files and external interfaces for the ECMS System. The Process Based Estimation analyzes each individual task derived from project scope and then uses Framework activity to estimate effort for the project.

11.2.1 Estimation technique *FP-Estimate*

Information Domain Value	**Estimated Count**	**Weight**	**FP Count**
Number of external inputs	12	x 4	= 48
Number of external outputs	9	x 5	= 45
Number of external inquiries	8	x 5	= 40
Number of internal logical files	9	x 10	= 90
Number of external interfaces	5	x 7	= 35
Count Total			= 258

Does the system require reliable backup and recovery? 3

Are data communications required?	5
Are there distributed processing functions?	3
Is performance critical?	3
Will the system run in an existing heavily utilized operational environment?	4
Does the system require on-line data entry?	2
Does the on-line data entry require input transaction to be built over multiple screens or operations?	1
Are the master files updated on-line?	3
Are the inputs, outputs, files, or inquiries complex?	4
Is the internal processing complex?	2
Is the code designed to be reusable?	1
Are conversion and installation included in the design?	1
Is the system designed for multiple installations in different organizations?	3
Is the application designed to facilitate change and ease of use by the user?	4
Total	**39**

11.2.2 Estimation technique *Process Based Estimate*

Activity --------->	**CC**	**Planning**	**Risk Analysis**	**Engineering**		**Construction Release**		**CE**	**Totals**
Task --------->				**Analysis**	**Design**	**Code**	**Test**		
List of ECMS Functions									
1. Identify need for engineering change	sadad			0.25	0.75	1.00	1.00		
2. Select and develop counteraction				0.25	0.75	1.25	1.00		
3. Specify document, track and decision change				0.50	1.50	1.75	1.00		
4. **Engineering implementation of change**				0.25	0.75	1.75	1.00		
5. **Manufacturing implementation**				0.25	0.75	0.75	1.00		

of change									
Totals	0.25	0.25	0.25	1.50	4.50	6.00	55.00		17.50
%effort	1.5	1.5	1.5	8.5	25.5	37.0	28.5		

11.2.3 Estimate for technique *FP-Estimate*

Function Points = 258 X [0.65 + 0.01 X 39] = 65.4
Estimation on how many lines of code (LOC) the software can now be calculated. Using a .Net language (ASP, C# or VB) we can calculate that for each function point there are 60 LOC, which boils down to a total of 3924.2 LOC for the entire program which is lower than expected but still sits in the 4000 LOC range.

Using the estimated LOC needed for the program we can now calculate person months (PM), Duration, and Staffing using the Semi-detached COCOMO model. Semi-detached is most relevant because our project is an intermediate (in size and complexity), software project in which we have members with mixed experience levels in which we have to meet a mix of rigid to less than rigid requirements. The following are the values the variables needed to calculate the relevant data.

"a" Variable	"b" Variable	"c" Variable	"d" Variable	KLOC
3.6	1.2	2.5	0.32	3.924

Effort (in PM) = $a*KLOC^{b} = 3.6*5.293^{1.2} = 18.56$
Duration (in Months) = $c*Effort^{d} = 2.5*18.56^{0.32} = 6.36$
Staffing = Effort/Duration = 18.56/6.36 = 2.9

These calculations show that it will take approximately 18.5 person months, 6.3 months, and 3 staff members to complete the software, however, with an extra staff member we can estimate that the duration of the development time is closer to 4 months than 6 months.

11.2.4 Estimate for technique Process Based Estimate

Using the Process Based Estimate, the person months needed to complete the project is 17.5, which is close to the function point estimate which was 18.5. Having effort, we can now calculate duration and staffing based on the semi-attached model.
Effort (in PM) = 17.5
Duration (in Months) = $c*Effort^{d} = 2.5*17.5^{0.32} = 6.25$
Staffing = Effort/Duration = 17.5/6.25 = 2.8

These calculations show that it will take approximately 17.5 person months, 6.3 months, and 3 staff members to complete the software, however, with an extra staff member we can estimate that the duration of the development time is closer to 4 months than 6 months. This matches the initial approximation of the functions point calculations.

11.3 Reconciled Estimate

The following estimate is an aggregate of the two estimates, averaging the two estimate techniques we can conclude that estimates are close to what we expect when we begin development.

Average Effort (in PM) = (17.5 +18.5) / 2 = 18
Average Duration (in Months) = c*Effort^{d} = 2.5*18$^{0.32}$ = 6.3
Average Staffing = Effort/Duration = 18/6.3 = 2.85

The cost of the project could vary. If we take a salary approach we can assume that the average salary of an entry developer is around $60,000. With four members we know that paying the cost of developers alone is $240,000. There are also other costs that are associated with the tools needed to complete the project. Those tools are: Visual Studios license x 4 ($400), and Microsoft Dynamic Nav license ($2500). The total cost of the project thus far is $242,900.

11.4 Project Resources

Preetinder Gill (client) and Dr. Maxim (professor) will be used as resources for the project. Subversion powered by Redmine will be the version control system we use as well as our repository. Microsoft Dynamic Nav[5] will be used to support out software. The software will be written in Visual Studios in a .Net language. Other resources needed are not yet determined.

12.0 Risk Management

The risk management is one of most important aspect of the project. In the following sub-section we will explore the potential risk we may face during this project. The risk will impact the project, product as well the business, in addition we are going to propose a strategy to manage, monitor and mitigate these risks.

12.1 RMM Introduction

The RMMM plan will involve several activities that focus on the risks that are most likely to occur and will have the most impact on the project. An action plan is put in place so that a formal process is used to avoid risks, if at all possible, or address risks when and if they occur.

12.2 Scope and intent of RMMM activities

After a list of risks is compiled, the lists will then be assessed based on likelihood of occurrence, and impact. After this is finished, the RMMM plan will be published. It will include useful actions on how to avoid the risk, manage and monitor potential risks. Risks that are deemed catastrophic will need to be closely monitored.

12.3 Risk management organizational role

Every member of the team will be responsible for Risk mitigation for the greater good of the project. Members will be mindful of the tasks being performed and the risks associated with those task. For example Project management tasks being poorly performed can result in scheduling issues.

[5] Due to budget restrictions, the ECMS team decided against using Microsoft Dynamic NAV module.

12.4 Project Risks

Data backup and security failure in Clouding computing	Product, Business	Data back up and its security is a concern in cloud
underestimated Performance & throughput requirements	Project and Product	The performance requirement may not be achieved
Data security failure	Product and Business	Security bridge due to a minimum security requirement
Unclear Software Architecture	Project, product	The architecture of the software is not well defined in the requirement - web-based or client server application?
Requirement change	Project and product	There will be a larger number of changes in the requirement than expected
Technology usage undefined	Project and product	How to use Microsoft Dynamic NAV to implement the software is not well defined
Size of company underestimated	Product and business	Mid-size business capacity underestimated
Technology or Tools failure	Product and business	Tools used during the implementation did not performed as expected
Product Competition	Business	Existence of competitive product in the market
Lack of Tool availability (Microsoft Dynamic NAV[6])	Project and product	Microsoft dynamic unavailable for exploration
Specification delay	Project and product	Lack of detailed information at start of the project will put the project plan behind schedule
Lack of data sample availability	Project and product	Inaccuracy in the use cases

12.5 Risk Table

The risk associated with development of software solution was studied. The critical risks have been presented in the table. (Probability and impact for risk m is described)

Risk	Probability	Cost	Strategy (Management)	Monitor (Avoid)	Mitigate	Impact (Effects)
Data backup and security failure in Clouding computing	Medium	High	*SaaS provider – Carbonite-major market	Data loss limited to one day only	Duplicate Copy of Data stored on other cloud computer or on local servers as	Serious

[6] Due to budget restrictions, the ECMS team decided against using Microsoft Dynamic NAV module.

					well	
Underestimated Performance & throughput requirements	Low	Medium	Make sure the major software performance requirements are met	NA	Engineer Change and IT are consult	Tolerable
Unclear Software Architecture	Low	High	A thorough investigation on software architecture before implementation	Any usual behavior of the software being watched regularly	Alternative architecture	Serious
Database performance	Low	Medium	Investigate the possibility of buying a higher-performance database	Frequent database performance report	Alternative of other storage solution	Tolerable
Size of company underestimated	Low	Low	Expand the capacity supported	Evaluate the growth of the company over time	Avoid unnecessary access to application	Tolerable
Underestimated development time	Medium	High	Investigate buying-in component and superior development tools	Check the project schedule regularly	Make adjustment in project schedule if necessary	Serious
Team member illness or absence	Low	High	Reorganize team to ensure cross-training	NA	NA	Serious
Data security failure	Low	High	Software solution with anti-virus compatibility and compliance	Security monitoring	High security network	Catastrophic
Technology usage undefined	Medium	Medium	Using similar technology	NA	Alternative technology that is similar	Tolerable
Requirement change	Medium	Medium	NA	Keep the client updated - frequently meet with the client for potential change(s) in requirements	Use adaptable technology	Tolerable
Specification delay	High	Medium	Make sure the client specifies all major software functionalities at start of the	NA	NA	Serious

			project			
Technology or Tools failure	Medium	High	Have a similar technology for backup - back up the data regularly	Closely monitor the technology used , duplicate the application and data	Use less complex technology - easy to troubleshoot	Catastrophic

Note: *SaaS (Software as a Service)

****Carbonite (Boston, MA)**

Carbonite provides online backup for small businesses and consumers by installing software that automatically backups up your files to the Internet. The product is priced annually per computer, with all plans providing unlimited storage. If the name sounds familiar to you, it is likely due to their national television marketing blitz that is pretty hard to miss.

12.6 Overview of Risk Mitigation, Monitoring, Management

RISK	MANAGEMENT	MONITOR	MITIGATE
Data backup and security failure in Clouding computing	SaaS provider –Carbonite-major market	Data loss limited to one day only	Duplicate Copy of Data stored on other cloud computer or on local servers as well
underestimated Performance & throughput requirements	Make sure the major software performance requirements are met	NA	Engineer Change and IT are consult
Unclear Software Architecture	A thorough investigation on software architecture before implementation	Any usual behavior of the software being watched regularly	Alternative architecture
Database performance	Investigate the possibility of buying a higher-performance database	Frequent database performance report	Alternative of other storage solution
Size of company underestimated	Expand the capacity supported	Evaluate the growth of the company over time	Avoid unnecessary access to application
Underestimated development time	Investigate buying-in component and superior development tools	Check the project schedule regularly	Make adjustment in project schedule if necessary
Team member illness or absence	Reorganize team to ensure cross- training	NA	NA
Data security failure	Software solution with anti-virus compatibility and compliance	Security monitoring	High security network
Technology usage undefined	Using similar technology	NA	Alternative technology that is similar
Requirement change	NA	Keep the client updated - frequently meet with the	Use adaptable technology

		client for potential change(s) in requirements	
Specification delay	Make sure the client specifies all major software functionalities at start of the project	NA	NA
Technology or Tools failure	Have a similar technology for backup - back up the data regularly	Closely monitor the technology used , duplicate the application and data	Use less complex technology - easy to troubleshoot

12.7 Catastrophic and Serious Risk Sheet

Risks	Severity of the Potential Impact	How to Reduce the Risk and Impact (detail RMMM)
Data backup and security failure in cloud computing	Due to its newness, cloud computing hosting is very challenging and difficult to control. Therefore, any failure of the cloud computer host can cause serious damage to the day-to-day business process	Duplicate copy of data stored on additional cloud computer or on local servers will provide an ultimate solution in case of failure of primary host
Unclear Software Architecture	Building the software on inappropriate architecture will definitely put the project in jeopardy	Conduct a thorough investigation on the appropriateness of the architecture to be adopted before the implementation stage
Underestimated development time	The development phase is the shortest in duration; however it is critical. Working with unfamiliar technology can quickly and easily prolong the development time.	Closely monitor the schedule plan, make sure to incorporate a CRP (critical time path), and include some delay triggers to alert all the team. Team members' familiarity with the newest technology prior to the implementation phase is also important.
Team member illness or absence	Team member absence can seriously affect the project, with the potential unfinished work or poorly executed tasks. Ultimately, this contributes to an incomplete product and unachieved goals.	This risk must be understood by all parties involved. An innovative approach to diminishing the impact, is to employ team cross-training during the course of the project. Multi-tasking and switching of roles by team members should be encouraged.
Data security failure	This risk is similar to the first issue "the security failure in the cloud"	All the security measures involving both software and hardware should be taken to assure excellent security.

12.8 Special conditions

- ***Trigger :*** Attempt to finish the step under intense time pressure :

  Experience shows a negative impact when we wait until the last minute to complete the work.

  Actions: Make sure all team members learned from past mistakes.

- ***Trigger:*** Frequent change in the requirement

  At the beginning the project, requirements will likely change. However, frequent change(s) in the requirement will cause the team to fall behind schedule; consequently the project will be in jeopardy.

  Actions: All parties involved, especially the client, should be aware of the consequences.

- **Trigger:** Frequent delay during the course of the project (running behind schedule in almost each step of project)

  During the course of the project, it is likely that we will run behind schedule from time-to-time. However, if this occurs consistently, the product will not be delivered on time

  Actions: follow the schedule strictly

13.0 Project Schedule

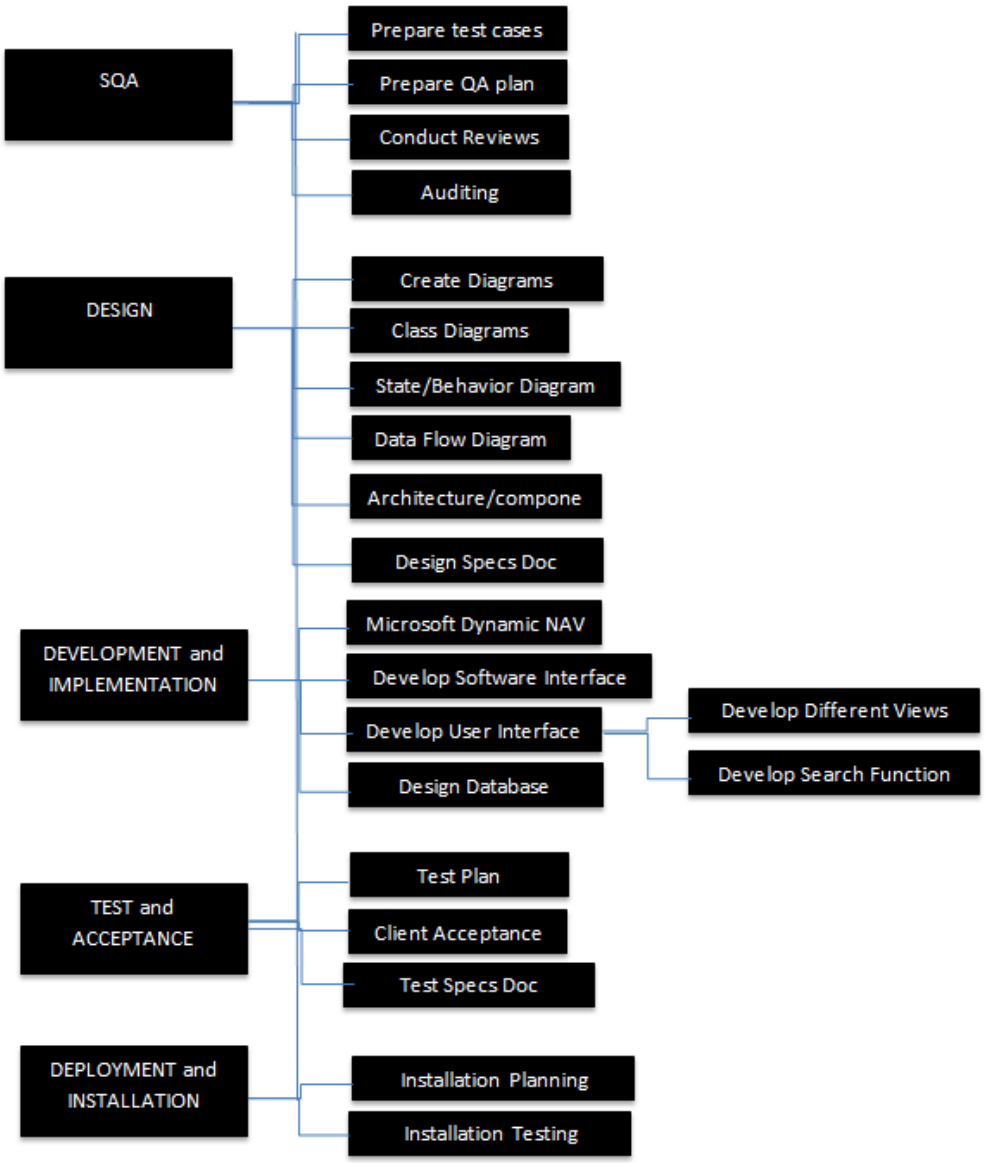
13.1 Project task set

Stage of Development	Deliverable	Deliverable Completion Date	Completion
Meeting with client		5/24/2012	x
Requirements	Requirements specification	6/5/2012	x
Project planning	Project Scope (inputs, functions, performance/behavior issues, Management and technical constraints)		x
	Project Estimates		x
	Risk Management plan		x
	Schedule		x
	Staff Organization		x
	Tracking and Control Mechanisms		x
	Milestone	**6/19/2012**	x
SQA	Define SQA tasks		x
	Create test cases and plans		x

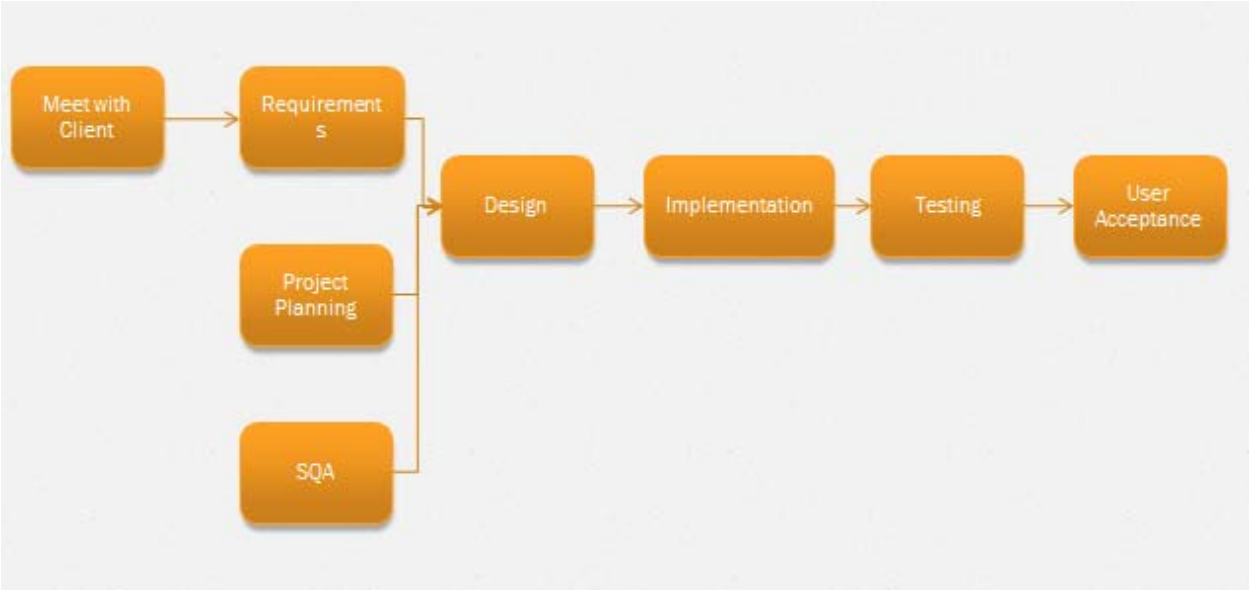
	Define Reviews and Audits		x
	Milestone	**8/14/2012**	x
Design	Program and Database Specs		x
	Data design		x
	Interface design		x
	Architecure and component level design		x
	Design Specification		x
	Milestone	**9/11/2012**	x
Implementation	Procure Software	9/30/2012	x
	Procure Hardware	9/30/2012	x
	Database Implementation		x
	Milestone: DB created		x
	Code login screen		x
	Code home/main screen		x
	Code Initiate Change Screen		x
	Code EC view screen		x
	Code MGMT approval screen		x
	View Search Screen		x
	View EC Summary Screen		x
	Milestone	**10/15/2012**	x
Testing	Test plan		x
	Test specs		x
	Unit Testing		x
	Integration Testing		x
	Milestone: Client Acceptance	**11/30/2012**	x
Installation	Project planning and doc		x

	Installation testing and verification		x
Final Documentation	**Milestone**	**12/12/2012**	x
			x

13.2 Functional decomposition



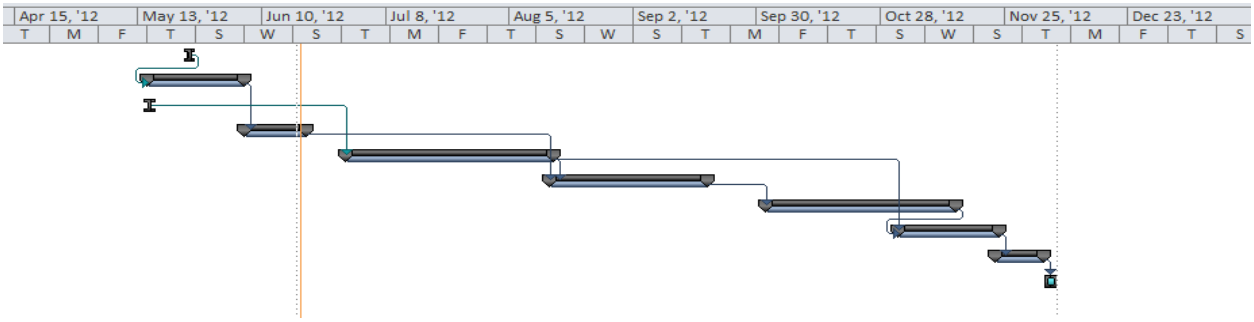
13.3 Task network



13.4 Timeline chart

Outline Number	Task Name	Duration	Start	Finish	Predecessors
1	Meet with client	1 day	Thu 5/24/12	Thu 5/24/12	
2	**Requirements**	**16 days**	**Tue 5/15/12**	**Tue 6/5/12**	**1**
2.1	Milestone: Req Doc				
3	**Project planning**	**10 days**	**Wed 6/6/12**	**Tue 6/19/12**	**2**
3.1	Milestone: Project planning doc	10 days	Wed 6/6/12	Tue 6/19/12	
4	**SQA**	**33 days**	**Fri 6/29/12**	**Tue 8/14/12**	**2**
4.1	Define SQA Tasks	11 days	Sat 6/30/12	Fri 7/13/12	
4.2	Define Reviews and Audits	11 days	Fri 7/13/12	Fri 7/27/12	
4.3	Milestone: SQA doc	3 days	Fri 8/10/12	Tue 8/14/12	
5	**Design**	**26 days**	**Tue 8/14/12**	**Tue 9/18/12**	**6**
5.1	Program and Database Specs	6 days	Tue 8/14/12	Tue 8/21/12	
5.2	Data design	3 days	Fri 8/17/12	Tue 8/21/12	
5.3	Interface design	3 days	Fri 8/17/12	Tue 8/21/12	
5.4	Architecure and component level design	18 days	Fri 8/24/12	Tue 9/18/12	
5.5	Milestone: Design Specification	3 days	Fri 9/14/12	Tue 9/18/12	

6	**Implementation**	**31 days**	**Tue 10/2/12**	**Tue 11/13/12**	**10**
6.1	Procure Software	7 days	Tue 10/2/12	Wed 10/10/12	
6.2	Procure Hardware	7 days	Tue 10/2/12	Wed 10/10/12	
6.3	Database Implementation	7 days	Wed 10/10/12	Thu 10/18/12	
6.4	Milestone: DB created			Thu 10/18/12	
6.5	Code login screen	2 days	Fri 10/19/12	Mon 10/22/12	
6.6	Code home/main screen	2 days	Fri 10/19/12	Mon 10/22/12	
6.7	Code Initiate Change Screen	7 days	Mon 10/22/12	Tue 10/30/12	
6.8	Code EC view screen	7 days	Tue 10/30/12	Wed 11/7/12	
6.9	Code MGMT approval screen	7 days	Tue 10/30/12	Wed 11/7/12	
6.10	View Search Screen	10 days	Tue 10/2/12	Mon 10/15/12	
6.11	View EC Summary Screen	3 days	Fri 10/26/12	Tue 10/30/12	
7	**Testing**	**17 days**	**Thu 11/1/12**	**Fri 11/23/12**	
7.1	Test plan	3 days	Thu 11/1/12	Mon 11/5/12	
7.2	Test specs	3 days	Thu 11/1/12	Mon 11/5/12	
7.3	Unit Testing	7 days	Mon 11/5/12	Tue 11/13/12	
7.4	Integration Testing	7 days	Tue 11/13/12	Wed 11/21/12	
7.5	Milestone: Client Acceptance	1 day	Fri 11/23/12	Fri 11/23/12	
8	**Installation**	**7 days**	**Fri 11/23/12**	**Mon 12/3/12**	
8.1	Project planning and doc				
8.2	Installation testing and verification				
9	Final documentation			Wed 12/12/12	



14.0 Staff Organization

The team consists of 4 developers, Janel Howard-Gumbs, Abdoul Razak Hamissou, Elisa Miller, and Arth Suthar. All members are seniors at the University of Michigan Dearborn collaborating with a real-world client from Bosch Engineering during the summer and fall semesters of the year 2012. The goal of the collaboration is to obtain applicable experience - taking a software design project from the requirements analysis phase through the

implementation and delivery of the product to the customer's site (prior to the end of the next semester). The team has a wide array of skills from project management, development, databases, and web design.

14.1 Team structure

The team structure for the project can be described as an open paradigm team in an agile environment; the team has been given complete autonomy to make decisions for the project, including technical decisions, financial/budgeting decisions, and decisions regarding scheduling. The only requirements that the team must conform to are those imposed by the business requirements of the client and classroom/academic requirements. The collaboration can also be described as democratic, with no one person with in the group having higher authority over other members.
Five major roles have been carved out to be taken on by all members at various times throughout the software lifecycle. These roles are; project manager, implementation analyst/technical writer, interaction designer, and developer/tester.

Project Manager Tasks include, but are not limited to:

A. defining project scope
B. defining project deliverables
C. creating the WBS (work breakdown structure)
D. defining and sequencing tasks for completing deliverables
E. resource estimation
F. time/cost estimation
G. scheduling
H. budgeting
I. risk planning and management
J. presenting packages to management for approval

Implementation Analyst/Technical Writer tasks include, but are not limited to:

A. preparing layouts for technical publication/submission
B. collaborating with clients and management to gather requirements
C. tracking and collecting reference material for project library
D. composing bibliography

Interaction Designer tasks include, but are not limited to:

A. gather client requirements
B. user analysis
C. user interface design and testing
D. usability testing
E. UI Prototyping (Wireframes)
F. Information Architecture Design

Developer/Tester tasks include, but are not limited to:

A. requirements gathering
B. programming
C. software installation and configuration
D. participating in coding and design reviews
E. database configuration

14.2 Management reporting and communication

All collaborators on the project have agreed to use common methods for internal communication (i.e internet/email) specifically Google Docs and Google Groups. The client and professor who are acting in

the capacity of management for this particular case have different methods by which reporting and communicating are facilitated. The client is informed via phone and email. The team reports to the Professor via email and VLT (Virtual Learning Tool) a web system that allows on campus and distance learners to access course material and interact with other students and professors.All deliverables are uploaded to the site for final review by management (the professor) on a bi-weekly rotation.

15.0 Tracking and Control Mechanisms

The team agrees to use some of the methodologies of PSP as a template, allowing for guiding conformance to software development standards and norms. Formal and Informal reviews and meetings will be the main source of tracking and control, along with other verification and validation activities such as code inspections, design reviews and product testing.

15.1 Quality assurance and control

Overview of SQA Plans Process steps will include scope and task review and refinement, defining and verifying operational specifications(See footnote in Appendix), defining and verifying Functional Specifications, (see footnote in Appendix), design review and verification, code audits and inspections, prototype testing, prototype inspection, and prototype verification, timely record keeping and reporting.

16.0 SQA Introduction

The SQA Plan defines the software team‘s SQA strategy. The mission of the SQA team is to assist the development team in producing a high-quality end product. The plan will include a Verification and Validation Plan, which describe how the verification and validation are carried out, as well as a Testing Plan, the primary SQA document used during the test.

16.1 Scope and intent of SQA activities

The overall objectives of SQA are the implementation of the following:

- **Requirements quality** (Analysis): The correctness, completeness, and consistency of the requirement model will have a strong influence on the quality of all work products that follow. SQA must assure that the software team has property reviewed the requirement model to achieve a high level of quality.

- **Design quality**: Every element of the deign model should be assessed by the software team to ensure that it exhibits high quality and that the design itself conforms to requirements. SQA looks for attributes of the design that are indicators of quality.

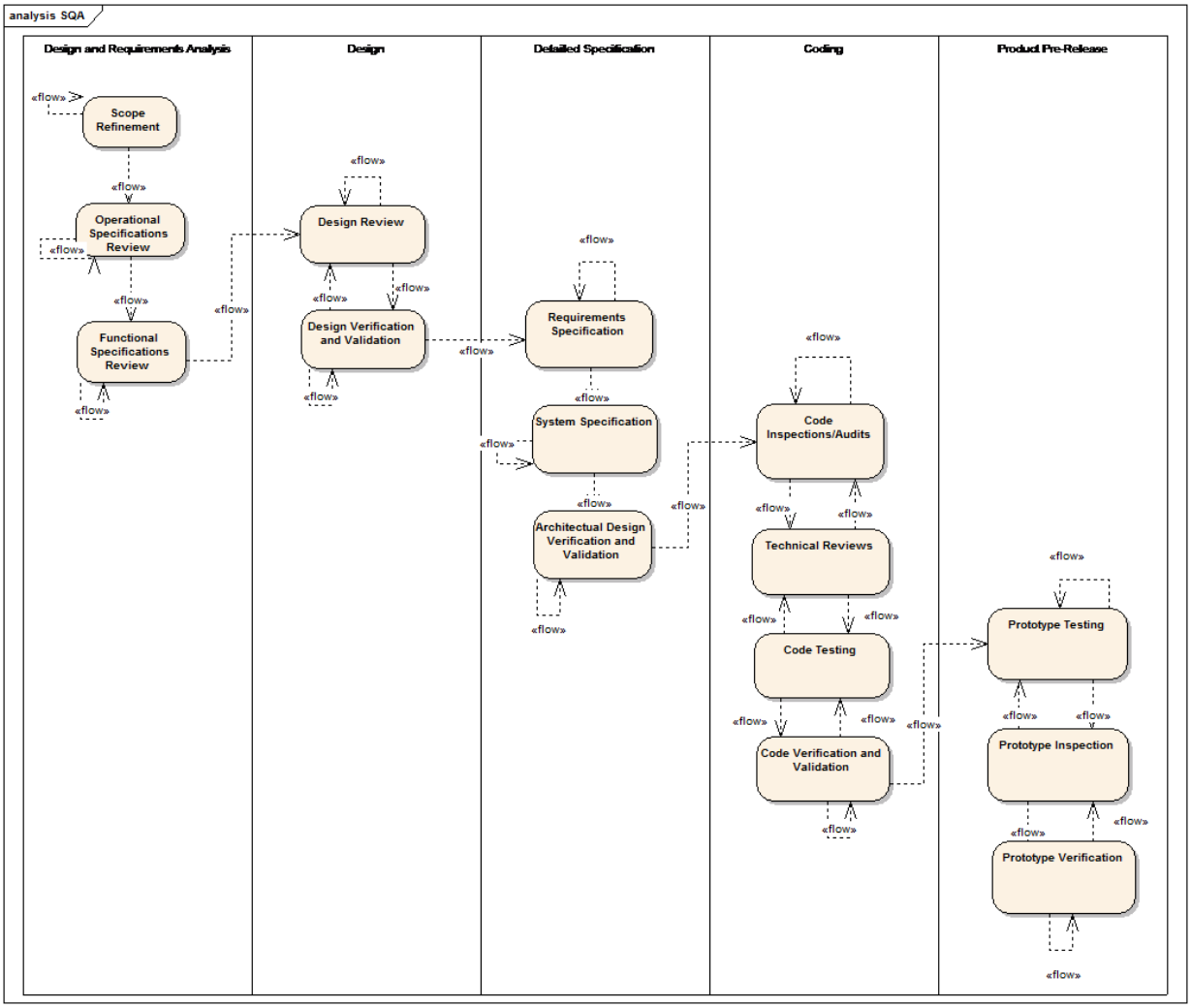
- **Code quality** (testing): Source code and related work product specification must conform to local coding standards and exhibit characteristics that will facilitate maintainability. SQA should isolate those attributes that allow a reasonable analysis of the quality of code.

- **Quality control effectiveness** (Pre-release testing and overall quality): The software team should apply limited resources in a way that has the highest likelihood of achieving a high–quality result. Ensure all the quality of the software functionality as whole before the release date fragment. SQA analyses the allocation of resources for reviews and testing to assess whether they are being allocated in the most effective manner.

All SQA major tasks are associated with two constituencies – the software engineer team, which is in charge of the technical work and the SQA group that is responsible for the quality assurance planning.

The project is composed of four students, and is divided into two groups for the purpose of SQA implementation; two students perform the responsibilities of the software Engineer team, and the two others members play the role SQA group.

The Scope of SAQ is as defined in the Road map of the previous Management Plan document (see diagram below).



16.2 SQA organizational role

- **Software Engineer** address quality by applying solid technical methods and measure, conducting technical reviews , and performing a well-planned software testing.
- **SQA group** is responsible for the quality assurance planning, oversight, record keeping, analysis, and reporting.

17.0 SQA Tasks

The SQA is composed of variety of tasks that are associated with the two constituencies; the software engineer team and the SQA group are responsible for the technical aspect of the project and the quality assurance planning, respectively.

17.1 Task Overview

- Prepares SQA plan for the project
- Participates in the development of the project's software process description
- Reviews software engineering activities to verify compliance with the defined software process
- Audits designated software work products to verify compliance with those defined as part of the software process
- Ensures that deviations in software work and work products are documented and handled according to a documented procedure
- Records any noncompliance and reports to senior management

17.1.1 Description of SQA task *m*

The following are the major tasks of the SQA:

- **Prepares SQA plan for the project**. The plan is developed as part of project planning and should be reviewed by all of the stakeholders. Quality assurance actions performed by the software engineer team and SQA group are governed by the plan. The plan identifies evaluations to be performed, audits and reviews to be conducted, standards that are applicable to the project, procedures for error reporting and tracking, and feedback provided by the software team.

- **Participates in the development of the project's software process description**. The software engineer team identifies a process for the work to be performed; the SQA group reviews the process description for compliance with organization policy, internal software standard, externally imposed standards (e.g., ISO 9001) and other parts of project plan.

- **Reviews software engineering activities to verify compliance with the defined software process**. The SQA group identifies, documents, and tracks deviations from the process and verifies that corrections have been made.

- **Audits designated software work products to verify compliance with those defined as part of the software process.** The SQA group reviews selected work products; identifies, documents and tracks deviations; verifies that corrections have been made; and periodically reports the results of its work to the project manager (Prof. Maxim and Prof. Akingbehin).

- **Ensures that deviations in software work and work products are documented and handled according to a documented procedure**: Deviations may be encountered in the project plan, process description, applicable standards, or software engineering work products.

- **Records any noncompliance and reports to senior management**: Noncompliance items are tracked until they are resolved.

17.1.2 Work products and documentation

The fruits of the SQA plan produced the following quality check tools:

- **Review**: process during which the content and status of the project is examined for error detection.
- **Audit**: When we compare actual practices to defined process or standard
- **Inspection**: A visual examination of a document with the primary goal of error or defect detection.
- **Assessment**: Self-study or third-party study to determine overall effectiveness and efficiency for possible improvement.
- **Quality Control**: Series of inspections, review audits, and tests throughout life cycle to ensure compliance to requirements.
- **Quality Assurance**: provides information about how the project is meeting quality goals and identifies problems that need correction.

The consequence of SQA implementation produces a **V&V Plan**, which describe how verification and validation are carried out, a **Test Plan**, which is the primary document used during testing, and a **Formal Technical Review Report**, the primary SQA product during analysis, design, and coding.

17.2 Standards, Practices and Conventions (SPC)

To assure that our product satisfies customer expectations by meeting the software specification, we are adopting the IEEE standard for SQA Plan and ISO 9001 Quality Standards.

17.3 SQA Resources

The personal resources of SQA are composed of a team of four students (SQA group and Software team). Through the course of this project, we used our personal computers for all the project work including SQA, and a variety of software and tools such as Windows OS, Microsoft Office 2010, Microsoft Visio, Enterprise Architecture, and SPC tools, previous project documents, lecture notes, books and Internet resources.

Reference: CIS 376 lecture notes, and Software Engineering (A Practitioner's Approach by Roger S. Pressman)

18.0 Reviews and Audits

18.1 Generic Review Guidelines

18.1.1 Conducting a Review

The formal technical review performed by the ECMS team will be an exchange of ideas regarding target work product. There will be completeness, equal say, the team will stay informed and we will do our best to defect detection.

18.1.2 Roles and Responsibilities

Each member of the ECMS team will hold a role so that the project is properly delegated. The roles will be: Investigator, Reviewer, and Recorder.

18.2 Formal Technical Reviews

18.2.1 System Specification Review

18.2.1.1 Description and Focus of Specification Review

The Specification document review will be conducted to figure out if the requirements of the ECMS project are correct. The focus of this review is to determine the correctness of our requirements in regards to the client's needs. The review will attempt to answer questions such as: Will the software specified by the requirements fulfill the client's needs? Is the software to be developed the right software for the client? If the requirements do not meet the client's needs then the software will be changed to fit what the client needs. If needed, the entire scope of the project will be reevaluated. In addition, the review will determine the amounts of resources needed. The software will also be reviewed to determine if the requirements contradict each other, conflicting requirements should be reevaluated to resolve the conflicts. Finally, the requirements that are finished should be reviewed for completeness.

18.2.1.2 Timing of the Review

The review will be conducted after the initial requirements gathering meetings with the client and once the requirements have been formally documented in the ECMS specification. The review will have a duration that is long enough to allow each area of the review to be completed while attempting to not surpass two hours. Furthermore, the software specification review shall occur before the planning phase of the development cycle commences.

18.2.1.3 Works Products Produced

Pending the finalization of the specification review, a final specification document that contains all the original data and new revisions will be produced as well as a completed review checklist.

18.2.1.4 Review Checklist

A checklist will be produced to finalize the review. The checklist will contain three criteria: correctness, completeness and scope. They will cover if the requirements are what the client wants, if the requirements are unfinished or not and if the requirements fit the ability of our development team.

18.2.2 Software Project Plan Review

18.2.2.1 Description and Focus of Project Plan Review

The project plan document will be reviewed so that the development team can complete the software successfully on time. The review will focus on key areas of the planning documents which include the project estimations, project schedule and team structure. Project estimates will help analyze if the project fits the scope of the project, while the schedule will help analyze if our development is done on time and there are not any conflicting schedules, and finally the team structure will help us delegate project responsibilities.

18.2.2.2 Timing of the Review

The review will be conducted after the specification document has been accepted and once the planning document is completed. The review will have a duration that is long enough to allow each area of the review to be completed while attempting to not surpass two hours. Furthermore, the software specification review shall occur before the planning phase of the development cycle commences.

18.2.2.3 Works Products Produced

Pending the finalization of the project planning review, a final project plan document that contains all the original data and new revisions will be produced as well as a completed review checklist.

18.2.2.4 Review Checklist

A checklist will be produced to finalize the review. The checklist will contain tree criteria: Estimations, schedule, and team management. They will cover if the estimates are in the scope of our project, if the correct time is allocated to project and if each member is performing their duties and are allocated sufficient responsibility.

18.2.3 Software Design Review

18.2.3.1 Description and Focus of Software Design Review

The software design document will be reviewed so that the software design and analysis match the software specifications. Design elements such as data design, architectural design, interface design and component design will be reviewed. The design review will help analyze if our choice of coding language, data types, structure of data transfer and overall interface is appropriate for the ECMS project. This will help determine issues such as process functionality and data structure correctness.

18.2.3.2 Timing of the Review

The review will be conducted after the project planning been accepted and once the software design review is completed. The review will have a duration that is long enough to allow each area of the review to be completed while attempting to not surpass two hours. Furthermore, the software specification review shall occur before the planning phase of the development cycle commences.

18.2.3.3 Works Products Produced

Pending the finalization of the software design review, a final design review document that contains all the original data and new revisions will be produced as well as a completed review checklist.

18.2.3.4 Review Checklist

A checklist will be produced to finalize the review. The checklist will contain tree criteria: analysis, design and user interface. They will cover if the analysis covers choice for design, if the analysis properly reflects the client's needs, if the design meets the programming language constraints, if the correct data structures are being used, and if the interface is easy enough for the targeted users to command.

18.2.4 Description of review *Code Review*

18.2.4.1 Description and focus of the review

The goal of the code review is to analyze source code for quality and correctness. Though multiple people will be coding the project, it should have a consistent feel, as if only one developer wrote the code in one sitting. Quality checks will include, but are not limited to, conformance to an agreed coding standard (i.e. white space, code readability, and naming conventions), unnecessary duplication of code, complexity, maintainability and narrow scope. Classes should be small with singular responsibility. The review will take place in a peer to peer forum. No automated tools will be used to assist in the review process.

18.2.4.2 Timing of the review

Code reviewed will take place on a bi-weekly basis, and all team members will be required to participate.

18.2.4.3 Work products produced

1. Completed check list :as described in 3.2.1.4

18.2.4.4 Review Code Review checklist

The following is an example of a script/checklist that will be used to perform a code review (taken from Karl E. Wiegers):

Structure

- Does the code completely and correctly implement the design?
- Does the code conform to any pertinent coding standards?
- Is the code well-structured, consistent in style, and consistently formatted?
- Are there any uncalled or unneeded procedures or any unreachable code?
- Are there any leftover stubs or test routines in the code?
- Can any code be replaced by calls to external reusable components or library functions?
- Are there any blocks of repeated code that could be condensed into a single procedure?
- Is storage use efficient?
- Are symbolics used rather than “magic number” constants or string constants?
- Are any modules excessively complex and should be restructured or split into multiple routines?

Documentation

- Is the code clearly and adequately documented with an easy-to-maintain commenting style?
- Are all comments consistent with the code?

Variables

- Are all variables properly defined with meaningful, consistent, and clear names?
- Do all assigned variables have proper type consistency or casting?
- Are there any redundant or unused variables?

Arithmetic Operations

- Does the code avoid comparing floating-point numbers for equality?
- Does the code systematically prevent rounding errors?
- Does the code avoid additions and subtractions on numbers with greatly different magnitudes?
- Are divisors tested for zero or noise?

Loops and Branches

- Are all loops, branches, and logic constructs complete, correct, and properly nested?
- Are the most common cases tested first in IF- -ELSEIF chains?
- Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
- Does every case statement have a default?
- Are loop termination conditions obvious and invariably achievable?
- Are indexes or subscripts properly initialized, just prior to the loop?
- Can any statements that are enclosed within loops be placed outside the loops?
- Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?

Defensive Programming

- Are indexes, pointers, and subscripts tested against array, record, or file bounds?
- Are imported data and input arguments tested for validity and completeness?
- Are all output variables assigned?
- Are the correct data operated on in each statement?
- Is every memory allocation deallocated?
- Are timeouts or error traps used for external device accesses?
- Are files checked for existence before attempting to access them?
- Are all files and devices are left in the correct state upon program termination?

18.2.5 RMMM review

18.2.5.1 Description of RMMM

This review focuses on risks that are most likely to occur and will have the most impact on the project. An action plan is put in place so that a formal process is used to avoid risks, or address risks when and if they occur.

18.2.5.2 Timing of RMMM

This will be conducted before the implementation phase.

18.2.5.3 Work products produced

- Reviewed issues List
- Summary report
- Meeting decisions
- Sign off sheet
- Accepted RMMM document
- Modification document

18.2.5.4 RMMM review checklist

- Review each of the current risks in risk table
- Have all plausible risks been assessed?
  Has anything changed?
- Category of risks correct?
- Impacts of risks correct?
- Is there corrective action for each risk?
- Recovery plans feasible or appropriate?
- Reprioritize risks?
- Are there new risks?
- Are there risks that need to be removed?

18.2.6 Test specification review

18.2.6.1 Description of Test specification

This review focuses on what types of testing were conducted and if they were able to discover as many defects as possible.

18.2.6.2 Timing of Test specification

This will be conducted after the implementation phase.

18.2.6.3 Work products produced

- Reviewed issues List
- Summary report
- Meeting decisions
- Sign off sheet
- Accepted Test Specification document
- Modification document

18.2.6.4 Test specification review checklist

- Test plan cover the entire functionality?
- Test plan include different types of software testing (unit, black, integration)?
- Test good enough to uncover defects?
- Will a new plan need to be written?
- Test new functionality?

18.3 SQA Audits

Scheduled software quality assurance audits will not be required for this project as all members of the ECMS will take part in the reviewing process. Additionally, the SQA activities audits carry a lower significance in this project as the development team will exist only for this project. Team members will not be required to reference the IAS SQA audits in future projects. However, it is important to note that effectiveness of both software engineering activities and software quality assurance activities will be discussed in regular team meetings along with other

project issues or concerns. Additionally, the team will discuss effectiveness of the SWE and SQA activities in the postmortem phase of the ECMS software development. During these discussions, each team member will have the opportunity to reflect upon the impact of the structure and activity results for bow SQA and SWE activities. The main objective of these discussions is to provide the ECMS development team members with insight as to improving a software process to yield even higher quality software.

19.0 Problem Reporting and Corrective Action/Follow-up

This section describes problem reporting mechanisms that occur as a consequence of the FTR's that are conducted and the means for corrective action and follow-up.

19.1 Reporting mechanisms

In the event a Formal Technical Review (FTR) surfaces an issue, the SQA Team will compile a report and email a copy of the report to the other team members. The report will also be filed in our report repository (DropBox).

19.2 Responsibilities

Each developer will be responsible and accountable for the code that he/she has written and will also own any revisions that should be made to the code. If an issue is uncovered by the SQA team on a particular section, then all members of the team will be informed and consulted to determine an action plan. Once an action plan is agreed to, it is the responsibility of the developer to edit the code to meet the team's agreed upon standards.

19.3 Data collection and evaluation

The following format will be used to collect and evaluate the data:

1. Reviewer Name
2. Date and Time of Review
3. Section of code reviewed with reference to LOC under review
4. Type of review performed
5. Description of Error/Defect
6. Type of Error
7. Error Rank (Minor, Moderate, Serious)
8. Action Plan for resolving
9. Sign Off by other team members

This will placed in table format in a word document and utilized in each FTR meeting. The updated document will be stored in DropBox and emailed to all parties. The results will be added to Pressman document appendix as needed.

19.4 Statistical SQA

Using the information from data collection an evaluation, each error can be categorized as one of the following types:

- Incomplete or erroneous specification (IES)

- Misinterpretation of customer communication (MCC)
- Intentional deviation from specification (IDS)
- Violation of programming standards (VPS)
- Error in data representation (EDR)
- Inconsistent component interface (ICI)
- Error in design logic (EDL)
- Incomplete or erroneous testing (IET)
- Inaccurate or incomplete documentation (IID)
- Error in programming language translation of design (PLT)
- Ambiguous or inconsistent human-computer interface (HCI)
- Miscellaneous (MIS)

Errors will be placed in a table in the following format:

	Total		Serious		Moderate		Minor
Error	No.	%	No.	%	No.	%	No.
IES	14	75	3	36	5	43	6
MCC	3	0	0	0	0	1	3
IDS	1	100	1	0	0	0	0
VPS	9	11	1	33	3	56	5
EDR	5	20	1	20	1	60	3
ICI	2	0	0	100	2	0	0
EDL	10	10	1	30	3	60	6
IET	11	9	1	18	2	73	8
IID	1	0	0	100	1	0	0
PLT	11	18	2	27	3	55	6
HCI	15	27	4	20	3	53	8
MIS	5	40	2	60	3	0	0
Totals	**87**	**18**	**16**	**30**	**26**	**52**	**45**

20.0 Software Process Improvement Activities

20.1 Goals and objectives of SPI

The main goal of SPI is to improve the software process in order to lower costs and develop a higher-quality software. We will look into:

1. Recording all errors and defects
2. Categorizing all errors and defects
3. Analyzing statistical information to determine which category has the most number or errors
4. Determine underlying cause of defects that occur
5. Record cost to correct each error
6. Develop a plan to change or tweak processes in order to reduce the frequency of errors in the category with the highest cost

20.2 SPI tasks and responsibilities

- Lessons learned meetings

21.0 Software Configuration Management Overview

SCM will be used to manage, identify and control software changes during implementation and testing. We want to limit the number of changes made, by eliminating unnecessary changes and ensure any changes made are implemented correctly.

Change requests will be communicated by email or word of mouth. If one of our members requests a change to be made, the whole team will review the change request. We will determine if the change is necessary and examine the impact the change would have on the project or software. If the majority of the team agrees to the change, then the person who requested the change will implement it. After implementation of the change, it will be tested to ensure that system performs as expected.

All change requests will be tracked and recorded, whether it is approved or denied.

Version control

The version number will be updated whenever a change is made to the system. We will use the standard x.x.x for version control:

- For minor changes: the hundredths digit will increase
- For substantial changes: the tenths place digit will increase
- For severe changes: the ones place will increase

22.0 SQA Tools, Techniques, Methods

Some techniques will be taken from the PSP (Personal Software Process) to help guide SQA Activities.

Some of the tools we will be using:

Office, Visual Studio, Drop box

Methods will include:

Basic code reviews

Team meetings

Problem tracking

Voting system

23.0 Change management and control

The team has chosen to use a version control system known as Subversion, which is powered by Redmine in conjunction with TortoiseSVN interface. The repository is located on a remote personal server.

24.0 Design Document Introduction

The project scope is composed of 5 categorized sequential processes: "**identification of engineering change"** process, which is trigger by the issue. The output of this process will either constitute the input of the "**selection and develop counteraction" process** if needed. Alternatively, it will trigger the next process, which is "**document specify, track and decision change**". Either of the aforementioned output processes can trigger the "**engineering implementation change"** process – its output can either be the cancellation of EC or "**manufactory implementation of change"**. The output of this process can either be implemented or canceled.

24.1 Goals and objectives

The Engineering Change Management Software (ECMS) process is categorized into the five following steps. The output of the previous process will constitute an input to the next process or the following one .It should also be noted that all the processes/activities are controlled by the organization's procedures.

1. **Identify need for engineering change**.

The mechanisms/users include: issue initiator, initiator's manager, customer(s) and supplier(s).

Input: the issue
Outputs: issue accepted as EC.
The accepted issues could be categorized as:

- needing development of counteractions,
- needing additional details for EC formalization
- accepted issue which doesn't need counteraction development or additional specification, but which is ready to progress to engineering implementation of change

2. Select and develop counteraction
The mechanisms/users include: issue initiator and initiator's manager.

Inputs: an accepted issue (that needs counteractions defined), and disapproved EC with recommendations for counteractions from the- **specify, document, track and decision change** process/activity.
Outputs: the recommendations counteraction, which are based on analysis by the EC team with respects to cost, time, quality, and system effects.

3. Specify document, track and decision change
The mechanisms/users include: issue initiator, EC team and action responsible.

Input: issue accepted as EC or recommended countermeasures
Outputs: an approved EC or a canceled EC. An approved EC is based on an evaluation from the EC team.

4. **Engineering implementation of change**
The mechanisms/users include: action responsible for engineering, supplier and customer.

Input: an issue accepted as EC or an approved EC.
Outputs: a released EC, an implemented EC or a canceled EC. Released EC entails issuance of a work order from engineering to manufacturing.

5. **Manufacturing implementation of change**
The mechanisms/users include: action responsible for manufacturing, supplier and customer.

Input: an issue accepted as EC or an approved EC.
Outputs: an implemented EC or a canceled EC.

24.2 Statement of scope

A description of the software is presented. Major inputs, processing functionality and outputs are described without regard to implementation detail.

24.3 Software context

This ECMS can be placed in various industries, ranging from engineering firms to medical practices. The software will reduce the time and money needed to verify background information, to plan implementation, to make decision and collaborate on engineering change tracking. This leads to more efficient handling of changes, happier customers and efficient use of company resources.

24.4 Major constraints

A web-based software solution, aimed to optimize engineering change process in order to improve the supply chain management in mid-sized organizations.

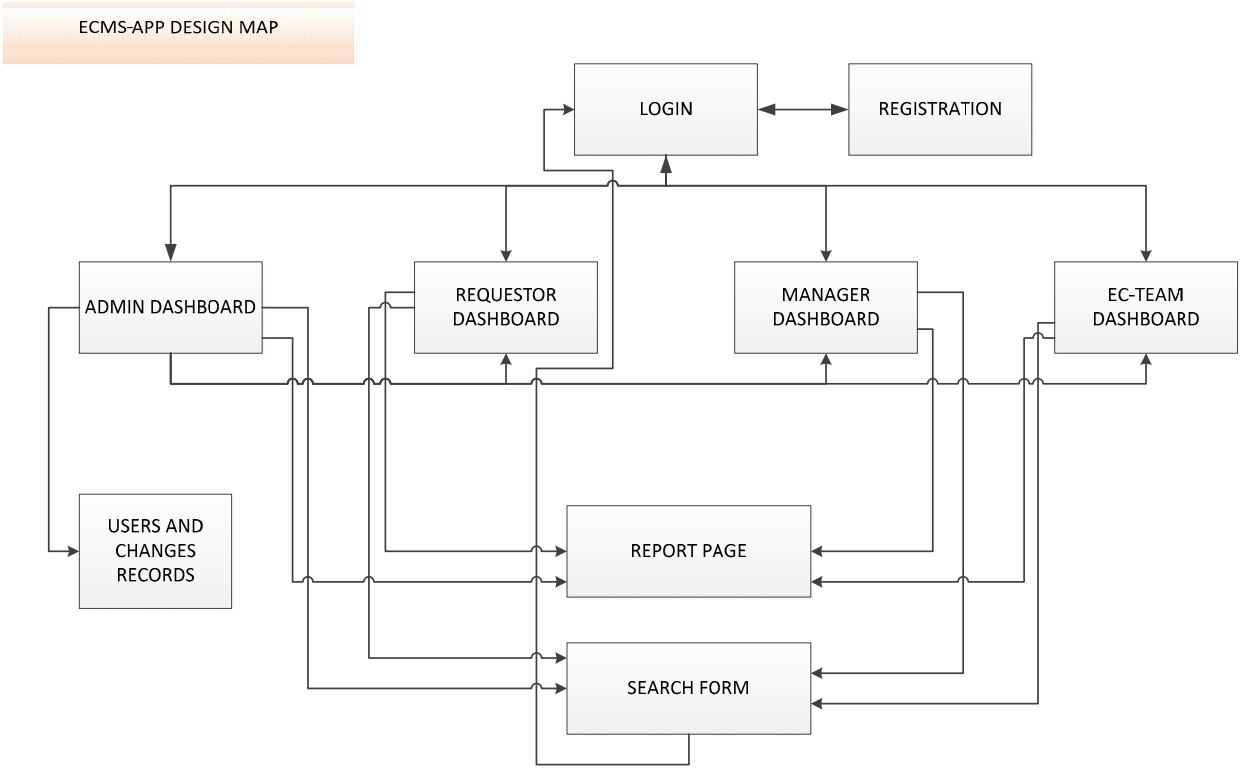
The software solution will be hosted on local onsite servers and PCs. The users should have predefined disk quotas. However, the software solution needs to backup all data to the cloud based data backup once in every 24 hours. The cloud based data backup is provided as SaaS by Carbonite. The engineering change identification numbers need to be communicated to an Oracle based ERP (using Sql Server) system through XML based data exchange. There need to be at least 50 user accounts, along with 5 administrative accounts and 1master account.

It should be able to run on Microsoft based 32-bit and 64-bit operating systems including but not limited to Windows Server 2008, Windows Server 2008 R2, Windows Vista Business, Enterprise, or Ultimate with SP1 or SP2, Windows Server 2003 SP2, Windows Server 2003 R2 SP2, Windows XP Professional SP3. Besides the local server, the maximum allocated PC hard disk space should be 1 GB. The memory requirements should not exceed 4 GB.

Sub components of the software need to be modular based on service oriented architectures (SOA service oriented architecture). Standard interfaces, such as XML, and protocols such as TCP/IP should be used to connect and link the software solution internally and externally.

25.0 Data Design

High level decomposition diagram



25.1 Internal software data structure

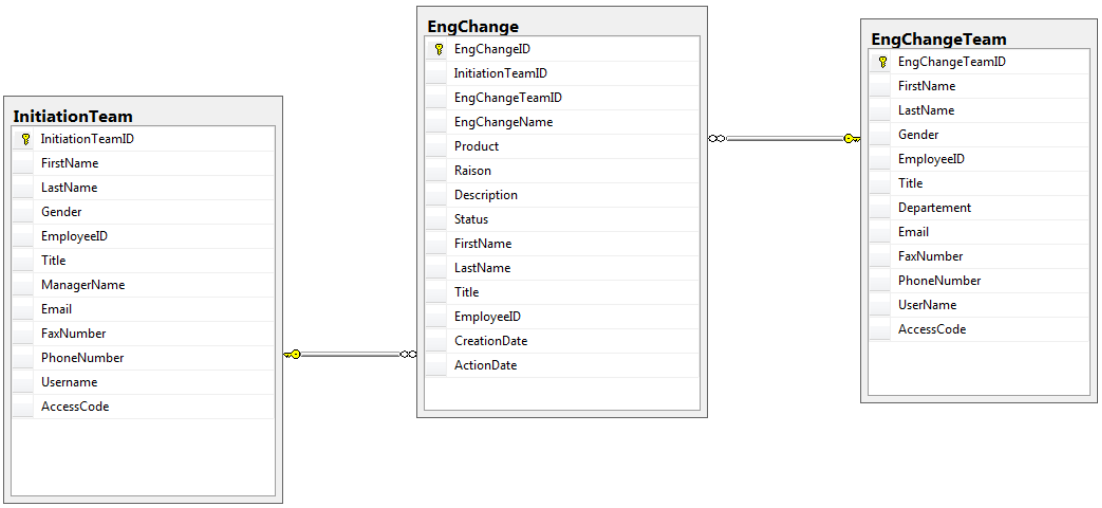
1. Data received/passed from User Login:
   - Username and password are authenticated for access against the company's internal security measures
   - User information is also stored in a session cookie to permit or restrict access to certain functions per privilege level
   - In the case of first time use – the user password will be stored inside the database
2. Data received/passed from ECMS Dashboard
   - The name of the specific engineering change that is input from the user will be selected from the database and returned to the system for display
3. Data received/passed from Administrator Tools
   - Option "Add Initiator"
   - Option "Add Engineer Change Team"
   - Option "Delete Initiator"
   - Option "Delete Engineer Change Team"
4. Data received/passed from Report Sheet
   - All reports that are available in the system are passed from the database to the system
   - The name of the selected report is sent from the client application and returned to the server
5. Data received/passed from Request Form

- User name, title, issue name, date, product, reason, and description are passed from the Request form to the system and stored in a log on the database

6. Data received/passed from Report Sheet “Quick Request”
    - Specific dates from the user are passed to the database and all requests that fall within the range are returned

25.2 Global data structure

Database Schema:



25.3 Temporary data structure

Not Applicable-No temporary data structure is needed for the ECMS System.

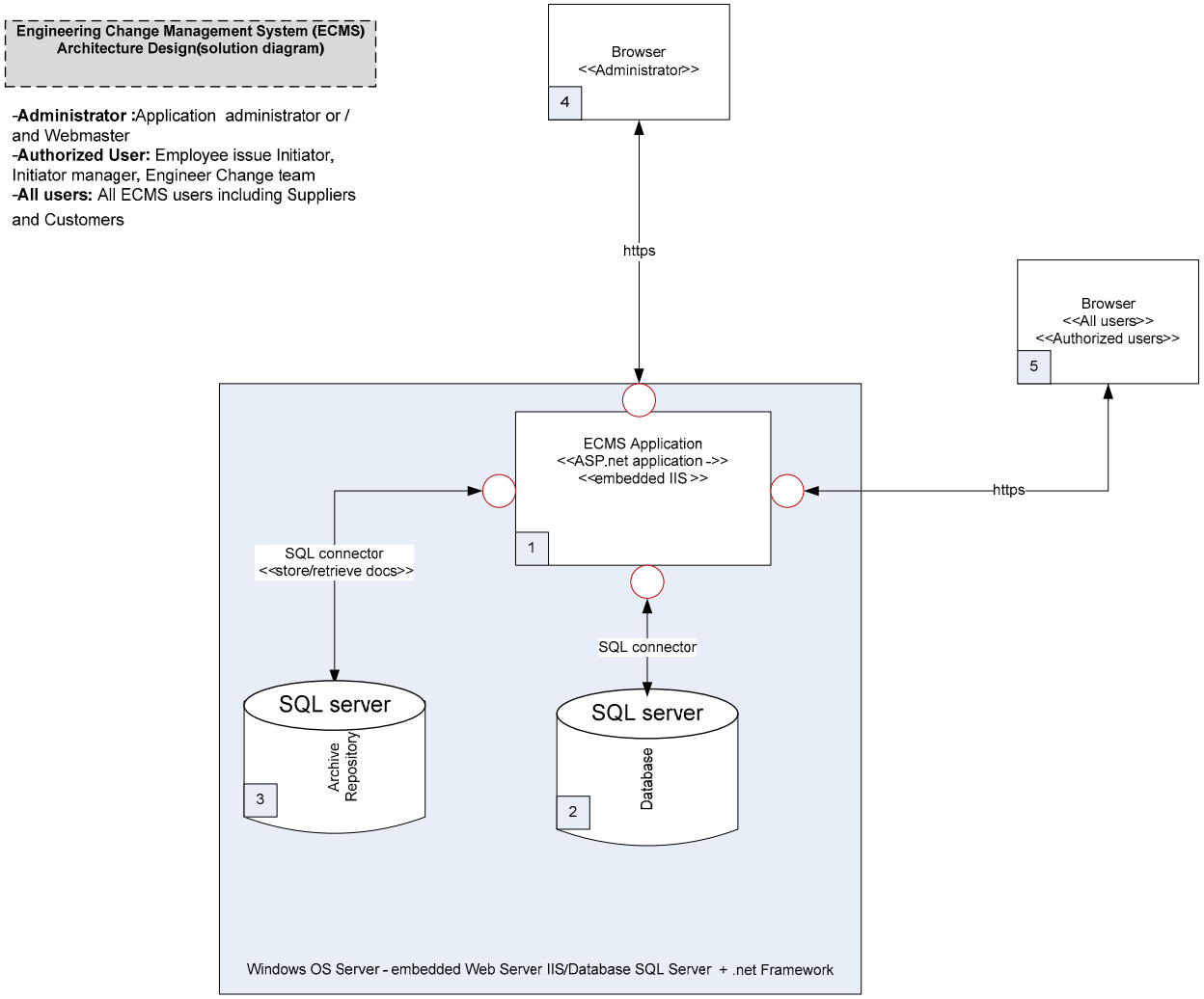
25.4 Database description

The database will consist of tables that will represent the structure in section 2.1. It will be hosted in a SQL Server database hosted by us. To enhance data security, all processing will be handled through executing stored procedures to prevent unauthorized users from manipulating the database. Passwords will be stored as hash codes.

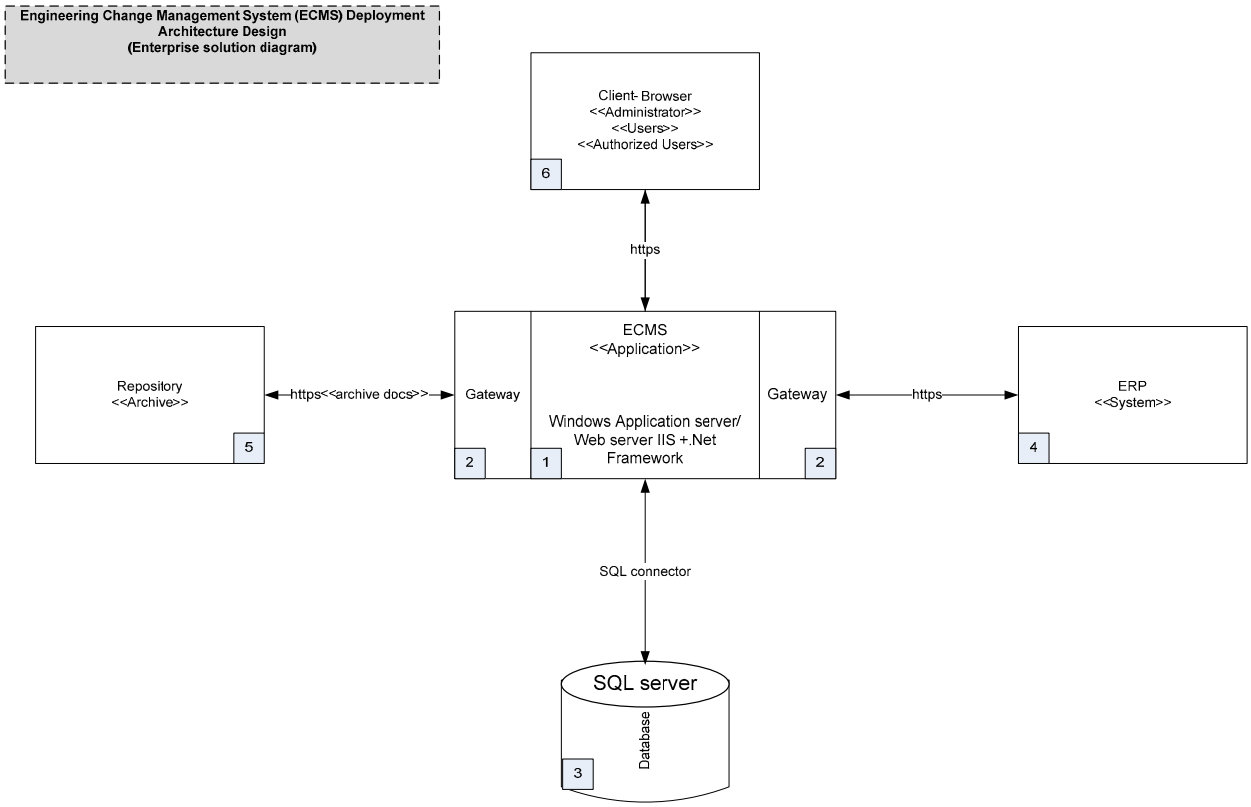
26.0 Architectural and component-level design

26.1 Program Structure Architecture diagram and Alternatives

Architecture Design – Prototype Testing



Architecture Design – Enterprise Deployment



26.2 Component Home/Login page

26.2.1 PSPEC for component for Home/Login page

The login page will be web based. All users: whether initiator, manager or EC team, will only be able to log in to the system through this page. It can be accessed from any browser. The info will be verified by the ECMS App (check the database).

26.2.2 Home/Login interface description

The screen will have two entry fields, one for user name and one for password. There will be a button "Log In", and a checkbox to remember the user. This screen will also display the ECMS logo on the top of the page.

26.3 Component Password Reset Form

26.3.1 PSPEC for component Password Reset Form

This page will be displayed when the user is logging in for the first time, or if the user wishes to reset the password, or it he user has forgotten the password.

26.3.2 component Password Reset interface description

This page will have 3 fields: one for existing password, one for new password, and one for confirming the new password that was entered. Two buttons: one for Change Password (submit), or one to cancel.

26.3.3 Algorithmic model for Home/Login and Password Reset

1. User fills in username/password
2. If the user is first time, or If user wishes to reset or change password
3. Prompt to enter password, and confirm password
4. Click “Change Password”

26.4 Component ECMS Dashboard

26.4.1 PSPEC for Component ECMS Dashboard

The page will be viewed by all users. This is where the user will be able to request a change, approve change, view a report, edit changes, update a status, search for a change. Administrative Tools will also be displayed here. Some functions will be restricted depending on the user, for instance, an initiator that is neither a manger nor part of the EC team, will be able to approve change.

26.4.2 ECMS Dashboard interface description

There will be several tabs displayed: Request Change, Approve Change, View Report, Edit Changes, Status Update, Search, and Administration Tools. Below the tabs a table of pending requests will be displayed.

26.5 Component Change Request Form

26.5.1 PSPEC for Subcomponent Change Request Form

This is the page (tab) that will be viewable by an initiator. This page allows a user to submit a request for review. Inputs include the name of the change, the product type, the reason for the change, the date, first and last name of the initiator, employee ID, and the title of the employee.

26.5.2 Component Change Request Form interface description

This page will have the title bar at the top displaying “Change Request Form”.

There will be 5 text boxes, each for: Change Name, Date, First name of employee, last name of employee, and employee ID.

There will be 3 drop downs, one for: Product type, reason, and title of employee.

There will also be a text box for the description of the change.

There will be two buttons, an ‘OK’ to submit the change, and ‘Cancel’ to cancel the request. The ‘Cancel’ will take the user back to the Dashboard.

26.5.3 Algorithmic model for Change Request Form

1. User selects Change Request Form Tab
2. User enters in all inputs: Change Name, Date, First/Last name, ID, Date, Product Type, reason
3. If user is satisfied with inputs
    a. Click Ok
    b. If not, click Cancel

26.6 Component Change Status Form

26.6.1 PSPEC for Component Change Status Form

This screen (tab) will be viewable only by the EC team. This page allows a member of the EC team to give or update a status on a change request that has been submitted. Inputs include the name of the change, the product type, the reason for the change, the date, first and last name of the initiator, employee ID, and the title of the employee, along with the status, status date, and creation date.

26.6.2 Subcomponent Change Status Form interface description

This page will have the title bar at the top displaying “Change Status Form”.

There will be 5 text boxes, each for: Change Name, Date, First name of employee, last name of employee, and employee ID.

There will be 4 drop downs, one for: Product type, reason, title of employee, and status

There will also be a text box for the description of the change.

There will be two buttons, an ‘OK’ to submit the change, and ‘Cancel’ to cancel the request. The ‘Cancel’ will take the user back to the Dashboard.

26.6.3 Algorithmic model for Change Status Form

1. User selects Change Request Form Tab
2. User enters in all inputs: Change Name, Date, First/Last name, ID, Date, Product Type, reason
3. If user is satisfied with inputs
    a. Click Ok
    b. If not, click Cancel

26.7 Component Change Status Form

26.7.1 PSPEC for Component Change Report View

This tab (screen) will only be viewable by managers. This will display all of the requests that have been submitted, with information about each status, such as who initiated the change, what the change is, the product, reason, a description, status, creation date and action date. This will also allow the user (manager) to edit the change if necessary.

26.7.2 Component Change Report View interface description

The page will have a title bar at the top displaying “Change Report View”

The list of requests will be displayed in table format. The table header will display **EngChange Name, Product, Reason, Desctipon, Status, First Name, Last Name, Title, Employee ID, Creation Date, Action Date.**

Underneath the header, the submitted requests will display. Next to each request will be a link "Edit" to allow a manager to edit a change if necessary.

26.8 Component EC Search Form

26.8.1 PSPEC for Component EC Search Form

This will allow the user to search for a change request, based on the criteria the user has inputted. This screen will be viewable by all users.

26.8.2 Component EC Search Form interface description

The page will have a title at the top displaying "EC Search Form"

There will be two text boxes: Initiator Employee ID and Date Create

Two drop-downs for: Product Type and Status

There will be two buttons: Search and Cancel

26.9 Component Administrator Tools

26.9.1 PSPEC for component Administrator Tools

This page (tab) will be an administrative page, allow the user to edit an Initiator account, EC account, or add a user to the database. The user will also be able to delete an account.

26.9.2 Administrator Tools interface description

There will be 5 buttons: Edit Initiator Account, Edit Engineering Change Team Account, Add Initiator, Add Engineering Change Team, and Close.

27.0 Component Edit Initiator Account

PSPEC and interface description for Component Edit Initiator Account

The list of users will be displayed in table format. There will be a header that displays **Initiation Team ID, FirstName, LastName, Gender, EmployeeID, Title, ManagerName, Email, Fax, and PhoneNumber.**

The users will be displayed below the header. There will be two links next to each user: Edit and Delete.

27.1 Component Edit Engineering Change Account

PSPEC and interface description for Component Edit Engineering Change Account

The list of users will be displayed in table format. There will be a header that displays **EngChangeTeam ID, FirstName, LastName, Gender, EmployeeID, Title, ManagerName, Email, Fax, and PhoneNumber.**

The users will be displayed below the header. There will be two links next to each user: Edit and Delete.

27.2 Component Add Initiator Account

PSPEC and interface description for Component Add Initiator Account

This page allows a user to add an Initiator account to the database. Inputs will be:

Six texts boxes for: First Name, Last Name, Email, Manager Name, Phone number, fax number

Radio button for Gender

Dropdown for Title

Two buttons: Ok and Cancel

27.3 Component Add Engineering Change Account

PSPEC and interface description for Component Add Engineering Change Account

This page allows a user to add an Engineering Change Team account to the database. Inputs will be:

Six texts boxes for: First Name, Last Name, Email, Manager Name, Phone number, fax number

Radio button for Gender

Dropdown for Title

Two buttons: Ok and Cancel

27.4 Software Interface Description

External machine interfaces

No other external machine interfaces will be needed for the ECMS system.

External system interfaces

Using client server architecture, the client will connect to company server through the internet using TCP/IP protocols.

Human interface

All human interfaces will are built as web interfaces using html and server side scripting language.

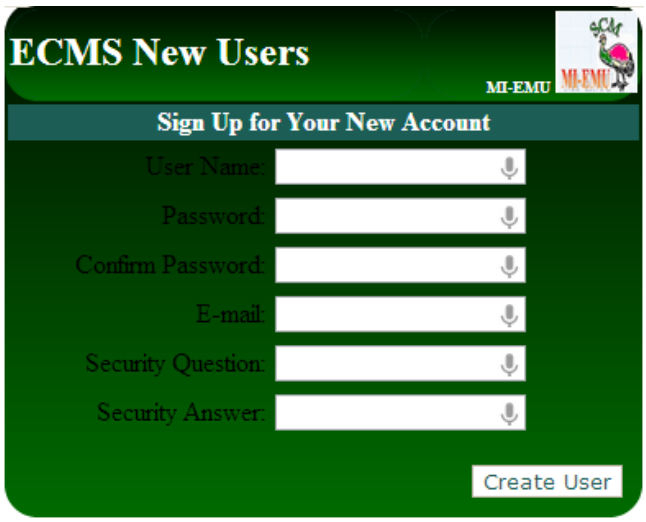
28.0 User interface design

28.1 Description of the user interface

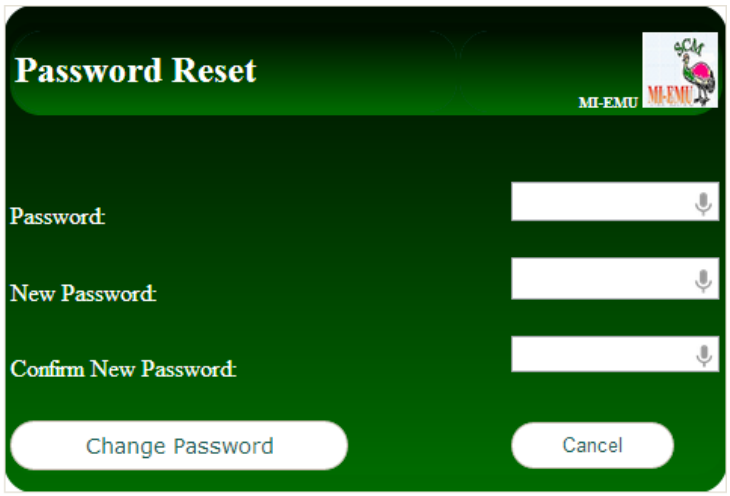
The following are screenshots and descriptions for several functions of the software that we have implemented in the GUI. Not all the functionality is ready yet, however, most of it is.

Below is the home screen, once the user has the browser pointed to the appropriate address (to be determined) the following login screen appears. Until authentication is completed the user may proceed unless the user would like to create a new account. (The following screenshots represent an earlier version of the application)

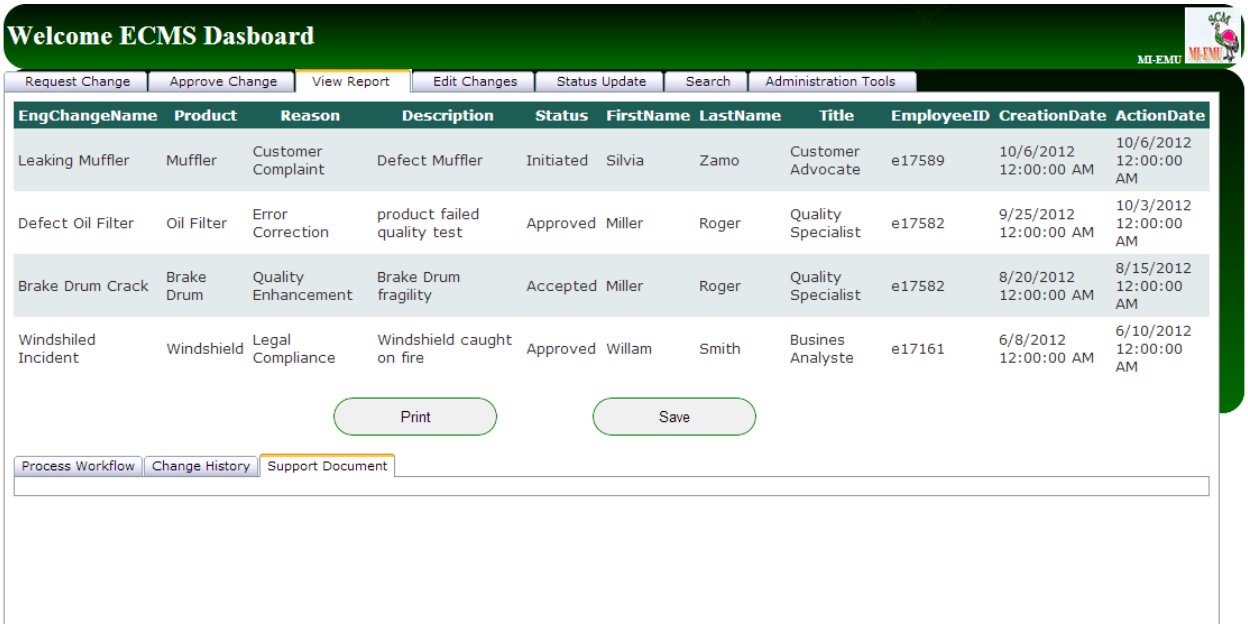
The “New User” screen lets a user create a new user provided they fill out the form correctly after creation they must go back to the Login screen and authenticate to proceed.



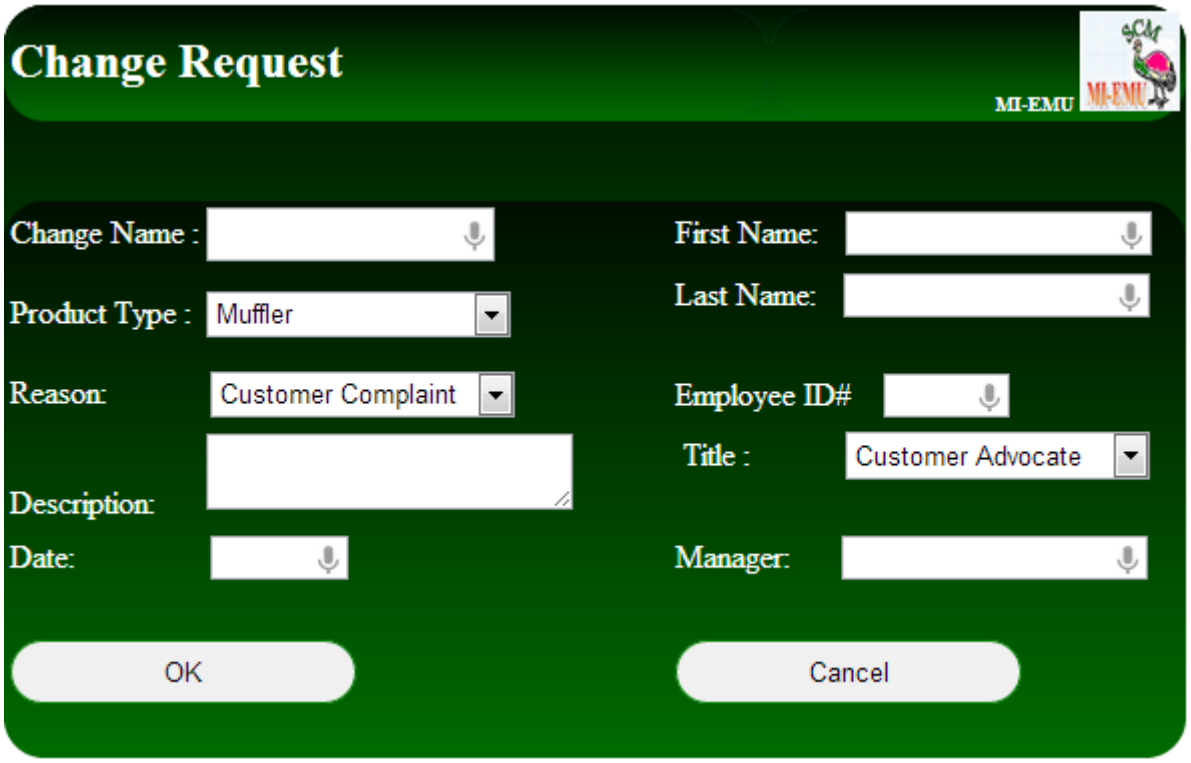
The following is a password reset form that allows the user to designate a new password for themselves provided they already know their original password.



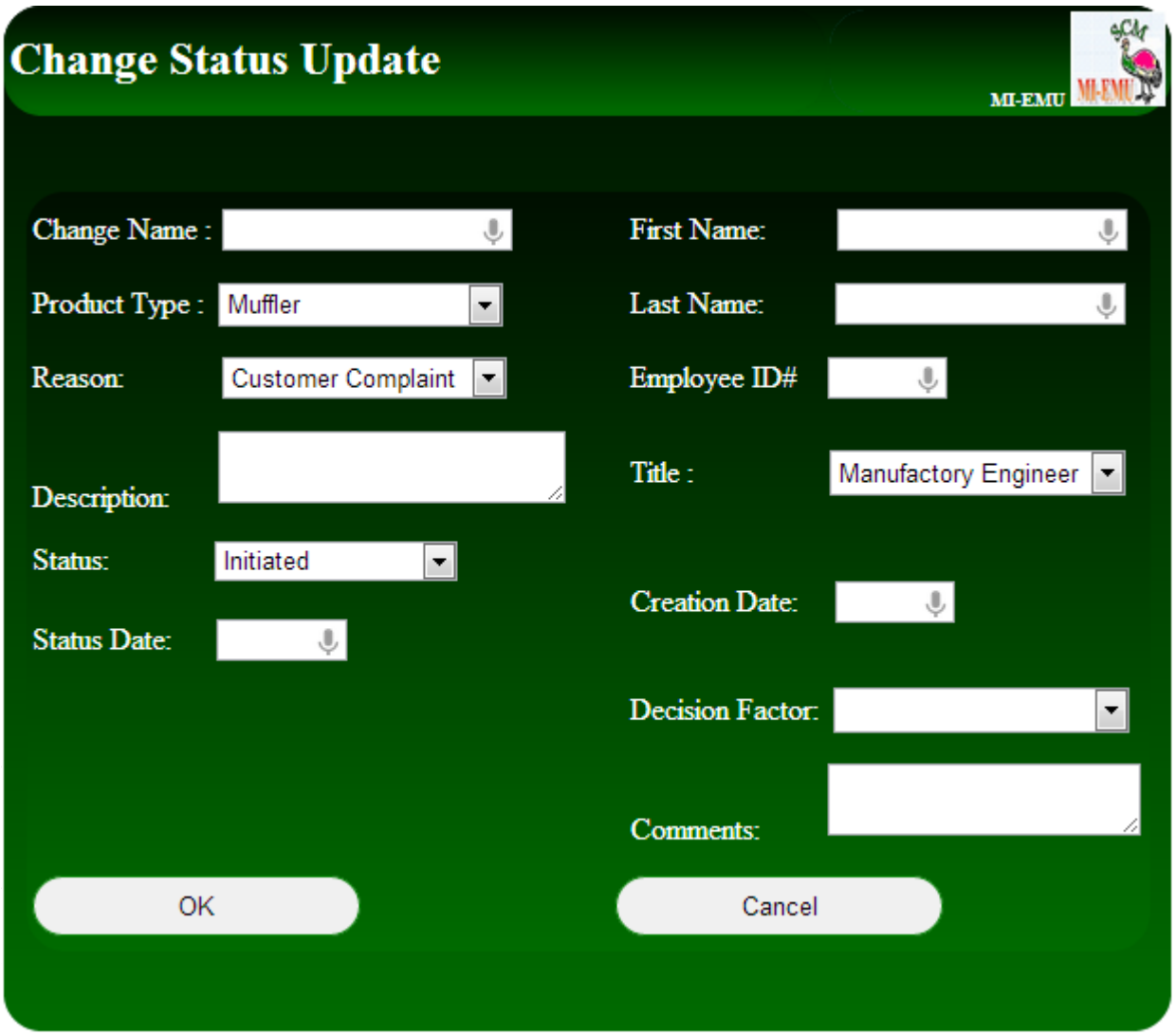
The dashboard screen lets a user view all the ECs and appropriate information based on the privilege level that user has.



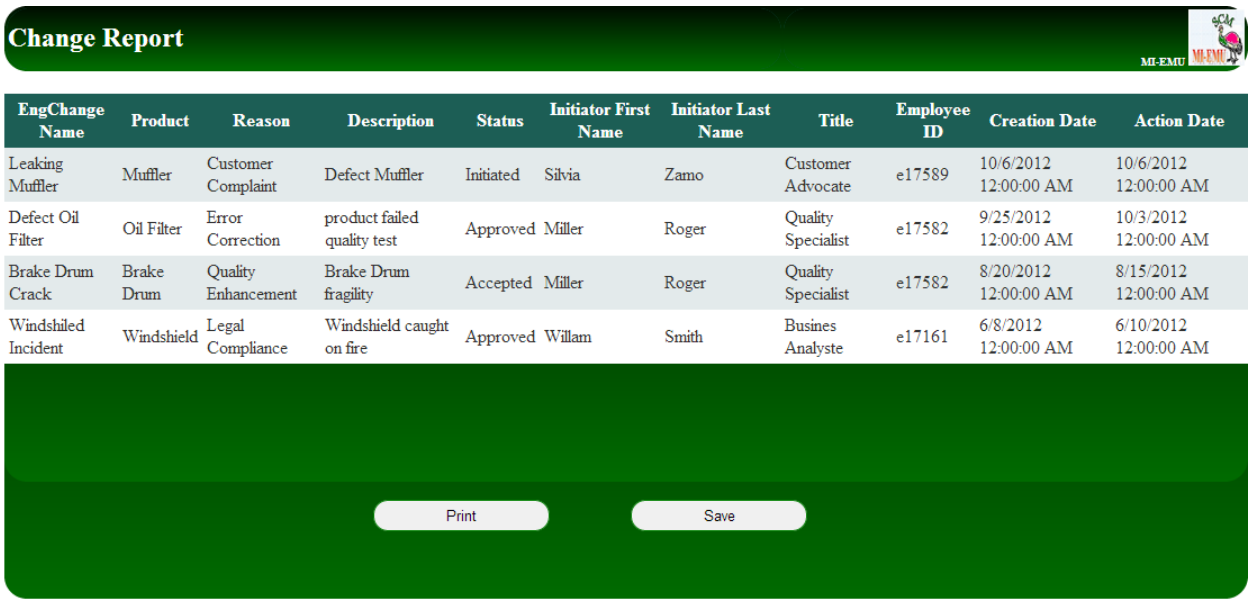
The change request form is filled by anyone who requires a change. Based on the type of product, reason and other pertinent information, the EC team can reject or accept a change. The change must first be requested, however.



Next is the Change Status Update, this allows a user Update information of an EC that has been already been created. The EC will be update and the pertinent information will show in the appropriate form.



The user may generate a report based and this report is printable. The following screen is the form that appears when a user wants to view or print a report.



Change Report

MI-EMU

EngChange Name	Product	Reason	Description	Status	Initiator First Name	Initiator Last Name	Title	Employee ID	Creation Date	Action Date
Leaking Muffler	Muffler	Customer Complaint	Defect Muffler	Initiated	Silvia	Zamo	Customer Advocate	e17589	10/6/2012 12:00:00 AM	10/6/2012 12:00:00 AM
Defect Oil Filter	Oil Filter	Error Correction	product failed quality test	Approved	Miller	Roger	Quality Specialist	e17582	9/25/2012 12:00:00 AM	10/3/2012 12:00:00 AM
Brake Drum Crack	Brake Drum	Quality Enhancement	Brake Drum fragility	Accepted	Miller	Roger	Quality Specialist	e17582	8/20/2012 12:00:00 AM	8/15/2012 12:00:00 AM
Windshiled Incident	Windshield	Legal Compliance	Windshield caught on fire	Approved	Willam	Smith	Busines Analyste	e17161	6/8/2012 12:00:00 AM	6/10/2012 12:00:00 AM

Print Save

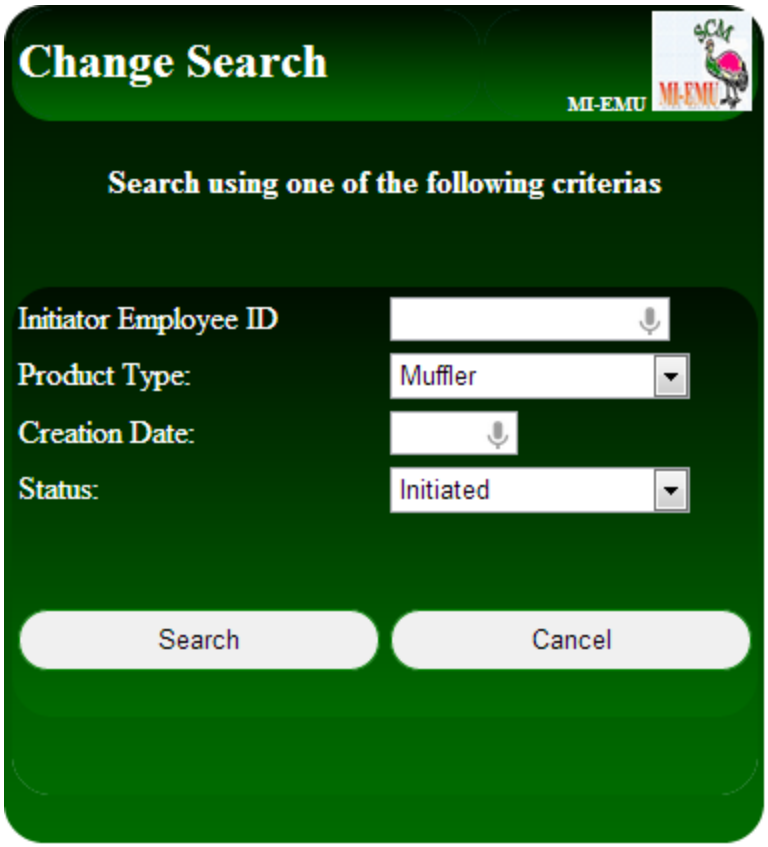
Next is a screenshot of the editable changes, based on the user's permission level they may be able to change and it's properties, from the name all the way to the status.

Change Edit

MI-EMU

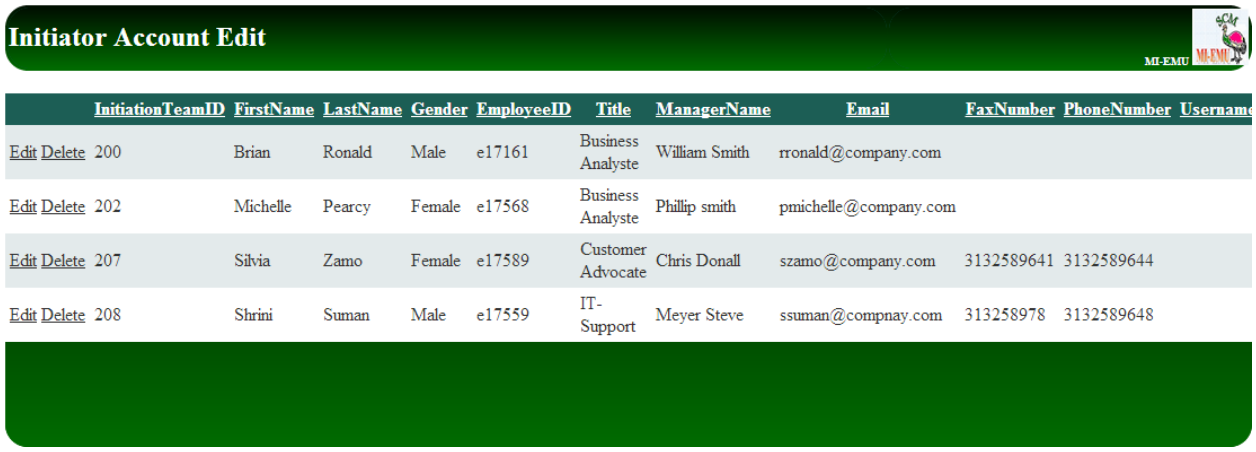
	EngChange Name	Product	Reason	Description	Status	Initiator First Name
Update Cancel	Leaking Muffler	Muffler	Customer Complaint	Defect Muffler	Initiated	Silvia
Edit Delete Select	Defect Oil Filter	Oil Filter	Error Correction	product failed quality test	Approved	Miller
Edit Delete Select	Brake Drum Crack	Brake Drum	Quality Enhancement	Brake Drum fragility	Accepted	Miller
Edit Delete Select	Windshiled Incident	Windshield	Legal Compliance	Windshield caught on fire	Approved	Willam

The change search form allows the user to search for a particular EC's and a filter is available to further narrow search results to get exact results.



The administrator tool panel is presented below. These are available only the site administrator and are powerful edits designed to make large changes.

The following is a screenshot of Initiator Account Edit form. This lets an administrator change the information of the initiator.



	InitiationTeamID	FirstName	LastName	Gender	EmployeeID	Title	ManagerName	Email	FaxNumber	PhoneNumber	Username
Edit Delete	200	Brian	Ronald	Male	e17161	Business Analyste	William Smith	rronald@company.com			
Edit Delete	202	Michelle	Pearcy	Female	e17568	Business Analyste	Phillip smith	pmichelle@company.com			
Edit Delete	207	Silvia	Zamo	Female	e17589	Customer Advocate	Chris Donall	szamo@company.com	3132589641	3132589644	
Edit Delete	208	Shrini	Suman	Male	e17559	IT-Support	Meyer Steve	ssuman@compnay.com	313258978	3132589648	

Next the Add Initiator Team is a form that can be accessed by the administrator panel. It allows for the administrator to create and add member to an initiator team.

Finally, we have a screenshot of the Add Engineering Change Team form. This is accessed via the administrator panel and is used to create and add members to the EC team.

There are several primary constraints to our system. Due to the timeline and size of our project, our final product is a prototype rather than a finished product. However, the prototype will be fully functioning with some minor features (that are not important to functionality) left out. We are also limited to a small budget. We must be able to complete the project with limited grant money and whatever our personal budgets allow for. This limits our ability to develop the project to exact specifications. Information of all EC's must be stored in a database and are only accessible by the software. Some constraints on the system will be the hardware on which the system is running. The speed of the system will most likely be determined by this factor. The final constraint is based on the how fast the network is; which determines the speed of the system.

30.0 Testing Issues

Some of the testing issues we ran into were lack of bug reporting in the repository, testing of separate entities: such as the database from the code, was hard to analyze due to complexity of the final product. Testing on highly secured networks – the ideal place for this product to be in was hard to do, due to the lack of security solutions available to us.

30.1 Classes of tests

Classes of Testing	Description	Expected Software Response	Performance Bounds
Login	Validate whether the user is able to login with his/her ID and password.	The user should be able to login when he/she enters a valid ID and password	The user will be denied access to the system if ID or password is invalid
InitiateChange	User can initiate the change	The user can start an EC	The user will not be able to create a change if they do not belong to the appropriate team and has not filled all the pertinent information
NotificationSent	Notification to the user is sent	Notification is sent to a user when changes are given	Notification is not sent to people who do not have access to that particular EC.
EditChange	User can edit the change	Allows the user to edit an existing EC	User edits are not permitted to those who do not have access
ECTeam	Test function and privilege level of the EC Team	Denies/Allows access to an EC based on privilege	Appropriate access is given to the correct user.
GeneralInitiator	Test function and privilege level of all eligible general initiators- these users should have restricted access and be unable to make major decisions on the engineering change (i.e.)	Restrict access to certain initiating users	Initiators may not have full access to the changes
ChangeLogged	Change Initiation and subsequent actions are logged in repository	Store log of all edits of all ECs	N/A
ReportGenerated	A printable report can be generated	Generate a printable report	N/A

30.2 Expected software response

- Incorrect username/incorrect password
- Comments fields should be populated
    - If field is empty, message should appear to put in comment
- Reject/cancelled with comments
    - Comment/reason field should be populated, or rejection will not be submitted
- EC submission should send out notification
    - Output that submission has been sent
    - EC team receives notification
- Status of change should send out notification
- Viewing historical data, invalid search produces no results, message should appear

30.3 Performance bounds

See “Classes of Tests” table from 6.1.

30.4 Identification of critical components

There are no critical components that demand particular attention during testing.

31.0 Supplementary Design Documents

31.1 Requirements traceability matrix

ID	*Functional Requirement*	*Status*	*Software Module(s)*	Tested In	Additional Comments
001	Initiate Change	Complete	InitiateChange()	VS & Browser	No Bugs
002	View Change	Complete	EditChange ()	VS & Browser	No Bugs
003	Approve Change	Complete	EditChange()	VS & Browser	No Bugs
004	Reject Change	Complete	EditChange()	VS & Browser	No Bugs
005	Cancel Change	Complete	EditChange()	VS & Browser	No Bugs
006	Edit Change	Complete	EditChange()	VS & Browser	No Bugs
007	Generate Report	Complete	GenerateReport()	VS & Browser	Bug Found. Fixed
008	Search Change	Complete	SearchChange()	VS & Browser	Bug found. Fixed.
009	Track Change	Complete	TrackChange()	VS & Browser	No Bugs
010	Notification of Change	Complete	NotificationSent()	VS & Browser	No Bugs
011	Validate Change	Complete	EditChange ()	VS & Browser	No Bugs
012	Login	Complete	Login()	VS & Browser	Bug found. Fixed.

31.2 Packaging and installation issues

There will be very limited packaging and installation issues because this software lives in the browser. We will not be installing or packaging the software on any particular machine.

31.3 Design metrics to be used

Metric	Description
Completion Rates	This metric allows users to record if their tasks are completed or not. They are recorded in a binary fashion, Task Success or Task Failure (1 or 0 respectively)
Usability Problems	Problems with the UI are recorded here. The problem must be described on both how many users encountered it and which users encountered it.
Task Time	Total task duration is used to measure how efficient and productive the design is. The metric records how long it takes a user to complete a task in seconds or minutes.
Task Level Satisfaction	Once a user has attempted a task we can ask the user to answer a few or a single question on how hard the task was. This allows for immediate detection of a difficult task.
Errors	This metric records any unintended mistake a user makes while attempting a task. This can help identify which tasks that are more frequently mistaken and may indicate that the function may need to be reevaluated
Page view/clicks	This is fundamental tracking metric which can measure efficiency of the application.
Single Usability Metric	Help track system tasks by combining metric into a single score. This can help understand the effectiveness of something like a dashboard.

32.0 Test Specification Introduction

The ECMS application is a browser based application composed of a login page followed by the ECMS Dashboard page. The ECMS dashboard serves as a home page for the application - giving users access to other features within the ECMS application. Access to the dashboard and subsequent pages is contingent upon registration to the site and the privilege levels of individual users.

32.1 Goals and objectives

Testing activities will need to:

- Verify that the software meets client expectations
- Verify the security of the software
    - User access is validated
- Verify that the software meets interface guidelines
    - Provide an intuitive environment for the user
- Expose bugs and uncover errors in the system.
- Validate any and all work products
    - Query Results
    - Reports

32.2 Statement of scope

Each tab within the application will serve as a stub for the test plan. Data that is input in each tab will need to be tested for validity, and invalid input must be communicated through the use of prompt boxes. Data that is passed from module to module will also need to be tested to verify that the input is properly placed in the database and correct results/software responses are displayed to the user either on screen or in crystal reports.

32.3 Major testing constraints

There are no major constraints that will impact the testing of the ECMS application.

33.0 Test Plan

33.1 Software (SCI's) to be tested

The software being tested is the ECMS, an online change management web application.

33.2 Testing Strategy

33.2.1 Unit Testing

Unit testing will be done on the lowest level. As an individual module is being written, testing will be conducted for each and every piece of functional code by compiling, reviewing and fixing compiler errors. When the coding stage of the module is complete, the module will then be tested according to its purpose. In this stage, the module will be tested with all reasonable and expected inputs to see if it does its expected task. In addition, the database and the associated modules will be tested specifically to locate any errors within our data. Finally, the module will be tested for security issues if applicable.

33.2.2 Integration Testing

After all individual modules and databases have been unit tested, and are properly functioning, they will be put together to form a complete application. This application will then be tested for general errors including compiler and runtime errors, input errors, and efficiency. Any errors will be corrected at this stage. If the program is inefficient in some computation, query or function, the cause of the inefficiency will be isolated and fixed, whether it be at the unit level or the integration level.

33.2.3 Validation Testing

Here we answer the questions “Are we building the right product?” As the application comes together, the programmers will make sure the software adheres to the client’s requirements. If a requirement is not met exactly, then the application will be modified to suit the most reasonable alternative.

33.2.4 Verification Testing

Here we answer the questions “Are we building the product right?” In this step, the group will make sure the software is being built correctly. We will evaluate the software to see if it is performing the appropriate functions according to the requirements and design specification.

33.2.5 High-Order Testing

The ECMS will be system tested to ensure that all programming modules integrate with each other as intended, and that the final software is integrated well into the native environment, the browser.

Alpha testing will follow, where the software interface will be tested by the entire team. The team will use the software as it was intended, to make sure no problems exist. This is where issues with the interface, databases, or graphs will be fixed.

The software will then be tested for security and performance. During security testing, we will attempt to run known exploits against the interface and test security of the software. Security is vital to a user interface, so as to keep user data confidential, and so any security issues will be given special attention. During performance testing we will test the runtime performance of the software and make sure that the interface performs well in a variety.

33.3 Testing Resources and Staffing

The ECMS is a web application that has user data running on a supported database. For testing the system, the database will be needed to test the modules. In addition, we will use the Visual Studios IDE to test the code itself and perform the other test once the application is completed using the appropriate browser. As far as personnel are concerned, each of the group members will be the ones testing the product after a prototype is constructed.

33.4 Test Work Products

Nothing will be produced through testing our application. Due to the nature of our software all “works” produced will be web pages.

33.5 Test Record Keeping

A test record keeping document will be used to evaluate immediate test result for each of the testing. In addition, a test log will be kept to monitor the tests that have been applied. An error, or “bug” log is kept to monitor any problems that have arisen during testing.

33.6 Test Metrics

The ECMS application will be tested to meet the quality and technical expectations proposed in the requirements specification. Usability, efficiency, reliability and completeness are a few important metrics that need to be accounted for during testing activity.

Usability – this will take into account a users' ability to navigate and use the software with ease. Although this can be applied after software builds, this metric will be taken into higher consideration at the final development stages of the product.

Efficiency – this metric will be used to measure the run time of the software during interactions between modules and interactions between the software and user.

Reliability – this metric will verify that functions will have consistent results after each operation.

Completeness – this metric will be used to test functions and the software as a whole, rating their functionality to the final versions specified in requirements.

33.7 Testing Tools Environment

The following testing tools will be used to evaluate our product:

- Visual Studios
- SQL Server
- ankSVN
- tortoiseSVN
- Redmine
- nUnit
- telerik
- Others to come as needed

Testing will be done within the browser, within the compiler, and on the database.

33.8 Test Schedule

The following is a tentative schedule; it may change if development is finished early or if development is delayed. It also should be noted that this does not include the schedule of testing done as we developed the product, however those tests are very informal.

Date / Test task	Week 13		Week 14	
Unit testing	→	→		
Integration testing		→	→	
Validation testing			→	→
Performance testing			→	→
Other testing				→

34.0 Test Procedures

34.1 Software to be tested

The ECMS web application

34.2 Testing Procedure

34.2.1 Unit Test Cases

Login Component

34.2.1.1 Stubs and/or Drivers

The stub will be the login screen.

34.2.1.2 Test Cases

- Test user that exists in the database
- Test user that does not exist in the database.
- Test blank input in username and/or password text box

34.2.1.3 Purpose of Tests

To ensure when a user logs in (successful or invalid) gets the appropriate message and response.

34.2.1.4 Expected results

A successful login will direct the user to the ECMS dashboard. Invalid login will display a message “Invalid user name or password”, or “Empty Login Field”. There should also be an option to register for access.

ECMS Dashboard Component

34.2.1.1 Stubs and/or Drivers

The stub will be the ECMS Dashboard screen.

34.2.1.2 Test Cases

- Test each tab (Request Change, Approve Change, View Report, Edit Change, Status Update, Search, Admin Tools, Process Workflow, Change History, Support Document)
- Test print and save buttons

34.2.1.3 Purpose of Tests

To ensure when a user clicks on each tab, he/she is directed to appropriate screen.

34.2.1.4 Expected results

- Clicking on Request Change Tab should display the Change Request Form
- Clicking on Approve Change should display Approve Change Form
- Clicking on View Report should display the View Report Screen
- Clicking on Edit Change should display Edit Change Form
- Clicking on Status Update Tab should display the Status Update Form
- Clicking on Search should display Search screen
- Clicking on Admin Tools should display the Admin Tools screen.

Requestor Form

34.2.1.1 Stubs and/or Drivers

The stub will be the Request Form screen

34.2.1.2 Test Cases

- Test all fields that are required, and drop-downs (First Name, Last Name, Product, Reason, Title, Date text, email, Change Name, Manager Name, Description)
- Test Submit and Cancel

34.2.1.3 Purpose of Tests

To test functionality of all fields, drop downs and buttons, and to ensure user fills in all fields that are required, and form is successfully submitted.

34.2.1.4 Expected results

If a user does not fill in a field that is required, an error will display next to the text box. If all fields are filled and user submits, the form should be successfully be submitted and added to the database. If the user clicks cancel, the user should be redirected to the ECMS dashboard.

Acceptor or Request Update Form

34.2.1.1 Stubs and/or Drivers

The stub will be the screen for updating a request.

34.2.1.2 Test Cases

- Test required fields, drop-downs (Update Date, Status, Comment)
- Test privilege level access to the update form. (not sure if this form should have access restrictions)

34.2.1.3 Purpose of Tests

To test functionality of all fields, drop downs and buttons, and to ensure user fills in all fields that required, and that the change update is successful submitted.

3.2.1.4 Expected results

- If a user does not fill in a field that is required, an error will display next to the text box. If all fields are filled and user submits, the form should be successfully be submitted and added to the database. If the user clicks cancel, the user should be redirected to the ECMS dashboard.
- If the user does not have the privilege level to make updates a message should be displayed “Access Denied – Proper Privilege Level is Required”.

Approval Form

34.2.1.1 Stubs and/or Drivers

The stub will be the Approval Form screen

34.2.1.2 Test Cases

- Test all fields that required, and drop-downs (Approval Date, Decision, Comment, Department)
- Test Submit and Cancel
- Test privilege level access to the approval form.

34.2.1.3 Purpose of Tests

To test functionality of all fields, drop downs and buttons, and to ensure user fills in all fields that required, and form is successful submitted.

3.2.1.4 Expected results

- If a user does not fill in a field that is required, an error will display next to the text box. If all fields are filled and user submits, the form should be successfully be submitted and added to the database. If the user clicks cancel, the user should be redirected to the ECMS dashboard. If the user does not have the privilege level to make approvals a message should be displayed “Access Denied – Proper Privilege Level is Required”.

Add Initiator Form

34.2.1.1 Stubs and/or Drivers

The stub will be the Add Initiator Form screen

34.2.1.2 Test Cases

- Test all fields that required, and drop-downs (Gender, Employee, PhoneNumber, FaxNumber)
- Test Submit and Cancel
- Test privilege level to the Add Initiator Form

34.2.1.3 Purpose of Tests

To test functionality of all fields, drop downs and buttons, and to ensure user fills in all fields that required, and the initiator is successfully added to the database.

34.2.1.4 Expected results

If a user does not fill in a field that is required, an error will display next to the text box. If all fields are filled and user submits, the initiator’s information should be successfully be submitted and added to the database. If the

user clicks cancel, the user should be redirected to the ECMS dashboard. If the user does not have the privilege level to add initiators a message should be displayed “Access Denied – Proper Privilege Level is Required”.

User Registration Form

34.2.1.1 Stubs and/or Drivers

The stub will be the User Registration Form screen

34.2.1.2 Test Cases

- Test all fields that required, and drop-downs (User Name, Password, Reenter Password, Email, First Name, Last Name, Role, Title, Department)
- Test Submit and Cancel

34.2.1.3 Purpose of Tests

To test password meets criteria, test functionality of all fields, drop downs and buttons, and to ensure user fills in all fields that required, to test input validation (email), and the user is successfully registered and added to the database.

34.2.1.4 Expected results

If a user does not fill in a field that is required, an error will display next to the text box. If all fields are filled and user submits, the initiator’s information should be successfully be submitted and added to the database. If the user clicks cancel, the user should be redirected to the ECMS dashboard.

User must also type password that meets criteria, and both fields must match. If they do not match or meet criteria, and error will be displayed next to the field.

Search Form

34.2.1.1 Stubs and/or Drivers

The stub will be the Search Form screen

34.2.1.2 Test Cases

- Test all fields that required, and drop-downs (Inititator Employee ID, Product Type, Creation Date, Status)
- Test Search and Cancel

34.2.1.3 Purpose of Tests

To test functionality of all fields, drop downs and buttons, and to ensure user fills in all fields that required, and the search is successfully performed.

34.2.1.4 Expected results

A progress bar will show as the search is being performed. A screen of results will display if something is found, if there are no results, a message should be displayed “0 results”

Report View

34.2.1.1 Stubs and/or Drivers

The stub will be the Report screen

34.2.1.2 Test Cases

- Test Print and Save Buttons
- Test that all fields of the report are properly populated

34.2.1.3 Purpose of Tests

To test functionality of all buttons

34.2.1.4 Expected results

- Clicking the print button should display the Print prompt.
- Clicking the save button should display the save prompt.

Administrator Tools

34.2.1.1 Stubs and/or Drivers

The stub will be the Administrator Tools screen

34.2.1.2 Test Cases

- Test each button (Edit account, Add account, Close)
- Test access to Admin functionality per privilege level

34.2.1.3 Purpose of Tests

To test functionality of all buttons

34.2.1.4 Expected results

- Clicking the Edit Initiator button should direct the user to the Edit Initiator form.
- Clicking the Add Initiator button should direct the user to the Add Initiator form.
- Clicking Close should redirect the user back to the dashboard

34.2.2 Integration Testing

34.2.2.1 Testing procedure for integration

We will integrate each component of the website in an incremental fashion and test each component as they are added. This way, errors are easier to detect, isolate, and correct. We will use a top-down approach, so a final test will be performed to ensure each component as been integrated correctly.

34.2.2.2 Stubs and Drivers Required

Stubs and drivers will be the same as Unit Testing and be performed again after each module or component is added.

34.2.2.3 Test cases and their purposes

Along with retesting each individual component after integration, we will have to ensure each hyperlink links correctly to the right page.

Normal tests – testing every condition is expected to occur with normal use of web app

Boundary test – testing every condition that may occur

Erroneous test – testing conditions that cause the system to time out

34.2.2.4 Expected results

As each module is integrated, the functions perform accordingly. There should be a seamless, error free product at the end of integration testing with all major modules tested and properly integrated.

34.2.3 Validation Testing

34.2.3.1 Testing procedure for Validation

Meet with the client to validate that our requirements meet his standards.

34.2.3.2 Expected results

Modules within the application must match the requirements that our client has given us.

34.2.3.3 Pass/Fail Criteria

Pass if the modules adhere to clients needs and ultimately if the client signs off on the final work product.

3.2.4 Verification Testing

3.2.3.1 Testing procedure for Verification

All team members will test the site as each role (initiator, admin, engineer team). We will each test and enact how the user would use the system. Functionality of each button and links will be tested.

34.2.3.2 Expected results

Valid inputs should submit form data or return search results, invalid inputs should output an error message. Each link should correctly link to the proper page.

34.2.3.3 Pass/Fail Criteria

Pass if the user receives output from the system regardless if the input is valid or not and pass if The modules must perform the appropriate function.

34.2.4 High-Order Testing

34.2.4.1 Recovery testing

Recovery testing will involve testing the ECMS web application in the event of a crash or when it is not functioning properly. If there is a crash or encounters and error, the system should display an error and notify the user that it needs to be restarted.

34.2.4.2 Security testing

The ECMS app requires a login whether the user is an initiator, part of the Engineering team, or admin team. An account must be created and password must meet certain criteria. If the user does not have the privilege level to perform various tasks within the ECMS system, a message should be displayed "Access Denied – Proper Privilege Level is Required".

- Incorrect Username and Password, access denied
- Correct Username and password, access granted and directed to the Dashboard
- Password validation, password must meet criteria and match
- Privilege level for access to approval and admin tabs must be verified

34.2.4.3 Stress testing

Some stress factors will be multiple users connecting to the application at the same time, the database is too small to hold the number of requests (and other data), memory of the server.

34.2.4.4 Performance testing

Speed will be a factor and a concern, and this will be based on the user's connection since the application is entirely web-based. Other factors will be the server that is hosting the application in terms of storage, processor speed, and memory.

34.2.4.5 Alpha/beta testing

Alpha testing will be performed by the development group and may be performed by fellow CIS students within the University of Michigan community as the testing phase moves closer to the release of the first prototype to the client. At that time, the team will determine any changes that need to be made before releasing a preliminary prototype to the client for beta testing.

34.2.4.6 Pass/fail criteria for all validation tests

Test Cases with Performance of Product on Test Plan

Test #	Test type	Description	Input	Expected Output	Actual Output	Pass/Fail
1	Security	Test user exists in the database	Username/password	Successful login – ECMS dashboard	Successful login of the software	Pass
2	Security	Test a user that does NOT exist in the database	Username/password	Message "Invalid User or Password"	Successful message displayed "Invalid User or Password"	Pass
3	Unit	Test blank	Nothing	Prompt user for user	Successful	Pass

		inputs on login screen		and password	prompt of user and password	
4	Unit	Test each tab menu of the ECMS dashboard	Mouse-events	Correct screen is displayed	Successful display of correct screen	Pass
5	Unit	Test all required fields	User inputs	Invalid inputs should result in error displayed	Successful display of error	Pass
6	Unit	Test privilege level access to update form	Mouse-event	If user does not have proper privilege level, message “Access is denied”	Successful display of message	Pass
7	Interface	Test correct display of pages	Mouse-Event	Pages are displayed correctly	Successful display of pages	Pass
8	Integration	Test Log In Button	Mouse-event	User directed to ECMS Dashboard	Successful direct to the ECMS Dashboard	Pass
9	Integration	Create User	User inputs (user info) and click “Register User”	User should be registered and added to database	Successful addition to the database	Pass
10	Integration	Password Reset	Matched passwords	Password changed successfully		
11	Integration	Password Reset	Mismatched password	Message “Passwords do not match, please try again”		
12	UI	Display of Report	Mouse-event	Reports displayed correctly	Successful display or report	Pass
13	UI	Save Report	Mouse-Event	Save prompt displayed		
14	Integration	Search Change that has been submitted	Submitted change, Employee ID, Product Type, Creation Date, Status	Result returned	Successful search of change	Pass
15	Integration	Search for a change that does not exist	Employee ID, Product Type, Creation Date, Status	Display Message “No records to display”	Successful display of message	Pass
16	Integration	Admin Tool Screen	Mouse-event (click on each button)	User directed to correct page	Successful direct of page	Pass
17	*Integration	Add Initiator Account	First name, last name, gender, employee ID, email, title, manager full name, phone number, fax number	Initiator account successfully created		
18	*Integration	Edit Account	Changing one of the fields of the account	Change for account should be updated		

19	Integration	Delete Account	Click Delete	User no longer exists in database	Successful deletion of account	Pass
20	White box + Security	Testing pages only users with certain privileges have access to	Click on tabs on Dashboard	“access denied” if user does not have privilege		
21	*UI	Approval Date text box, must be higher than Request Date and Acceptance Date	Correct date, incorrect date	Correct date allows submission, incorrect date displays “Date cannot be higher than Request and Acceptance Date”	Error occurred when pulling data	
22	*UI	Comment box on Accept/Request Update Form	Comment, no comment	No comment displays “Please enter in comment”		
23	Integration	Submit Change	Change Name, Product Type, Reason, Description, Date, First Name, Last Name, Employee ID, Title, Manager	If all fields are filled correctly, change should be submitted. If there are fields that are left blank, message displays “all fields required”	Successful submission of change	Pass
24	Integration	Reject Change	Mouse event	The change is rejected	Successful rejection of change	Pass
25	Integration	Search Reset	Mouse event	Field in the search form are reset	Successful search field reset	Pass

34.3 Testing resources and staffing

Preliminary testing will be done by each member on the development team using Microsoft Visual Studio ASP.net development tools. Once, the application is migrated from local PC’s to a hosted server. Each developer will have access to the server using an SSH client.

34.4 Test work products

No formal work product will be produced as a result of the testing phase. Some informal work products may include mock reports generated with sample data and screenshots to verify application response.

34.5 Test record keeping and test log

All work products will be stored in the ECMS dropbox folder and placed in a log that will include the name of the tester, date of the test, component to be tested, type of test, reason for test, expected results, and a description of

results. If expected results are not achieved; a detailed action plan should be determined by the team to correct the issue. Subsequent testing should be performed to achieve expected results.

The following is a test log of how we are going to keep track of bugs during testing.

Test Log					
Date	**Defect report**	**Test Type**	**Test Case**	**Result**	**Comments**
10/16	Login Credentials not working	Security	2	Password criteria changed	Changed our authentication method
10/16	No prompt for invalid user name/password	Unit	3	Prompt added to login	
10/16	Wrong Tab Redirect	Unit	4	Changed the link to appropriate page	
10/16	Invalid Email format still worked	Unit	5	Corrected email format	
10/20	User not being added to the database	Integration	9	Users now correctly added to database	
10/20	Password Change	Integration	10	Passwords were not being changed	Changed with new authentication method
10/20	Change Record Not Populating	Integration	N/A	Records are now populated correctly	
10/21	Display of Reports	UI	12	Report divs not displaying correctly	
11/11	Approval Date Mismatch	UI	21	Correctly Approves Date	
11/20	Comment box not requiring input	UI	22	Now requires data in field to proceed	

Errors and bugs will be also be tracked via the repository. The repository we are using is powered by Redmine and it offers a diverse set of record keeping such as: bug tracking, activity logs, roadmaps, and milestone succession.

35.0 Appendix

35.1 Code listing

Default.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Configuration;
using System.Data.SqlClient;
using System.Web.Security;


public partial class ECMS_Login : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Session["ecmsRole"] = null;
    }
/*-------------------------------------------------------Authentiaction and
Redirection-------------------------------------------------------*/
    protected void SubmitButton_Click(object sender, EventArgs e)
    {

        // set session state
        string username = UserName.Text;
        string selectSQL = "SELECT Role FROM Employee WHERE UserName = '" + username +
"'";
        string selectSQL2 = "SELECT UserName FROM Employee WHERE UserName = '" + username
+ "'";
        string selectSQL3 = "SELECT Password FROM Employee WHERE UserName = '" + username
+ "'";

        SqlConnection con = new
SqlConnection(ConfigurationManager.ConnectionStrings["ECMS_DatabaseConnectionString"].Con
nectionString);
        SqlCommand cmd = new SqlCommand(selectSQL, con);
        SqlCommand cmd2 = new SqlCommand(selectSQL2, con);
        SqlCommand cmd3 = new SqlCommand(selectSQL3, con);
        con.Open();

        string ecmsRole = (string)cmd.ExecuteScalar();
        string chkUsername = (string)cmd2.ExecuteScalar();
        string chkpassword = (string)cmd2.ExecuteScalar();

        Session.Add("ecmsRole", ecmsRole);

        if (chkUsername == null )
        {
            Label1.Visible = true;
            Label1.Text = "Invalid UserName!";

        }
        else if (chkpassword != Psswrd.Text)
        {
```

```
            Label1.Visible = true;
            Label1.Text = "Invalid Password!";

        }
        else if (Session["ecmsRole"] == null)
        {
            Response.Redirect("nologin.aspx");
        }
        else if (ecmsRole == "Requestor")
        {
            Response.Redirect("RequestForm.aspx");
        }
        else if (ecmsRole == "Manager")
        {
            Response.Redirect("ValidationForm.aspx");
        }
        else if (ecmsRole == "EC-Team")
        {
            Response.Redirect("ApprovalForm2.aspx");
        }

        else if (ecmsRole == "admin")
        {
            Response.Redirect("AdminDashboard.aspx");
        }

        }


}
```

AdminDashboard.aspx

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;

public partial class AdminDashboard : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["ecmsRole"] == null)
        {
            Response.Redirect("nologin.aspx");
        }
        else if (Session["ecmsRole"].ToString() != "admin")
        {
            Response.Redirect("notauth.aspx");
        }
    }
}
```

ApprovalForm2.aspx

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Configuration;
using System.Data.SqlClient;

public partial class ApprovalForm2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!this.IsPostBack)
        {
            FillChangeInfo();
        }
        //InitializeCulture();
        if (Session["ecmsRole"] == null)
        {
            Response.Redirect("nologin.aspx");
        }
        else if (Session["ecmsRole"].ToString() != "EC-Team" &&
Session["ecmsRole"].ToString() != "admin")
        {
            Response.Redirect("notauth.aspx");
        }
    }


    //Populate form form request data
    private void FillChangeInfo()
    {
        DropDownList2.Items.Clear();
        string selectSQL = " SELECT Industry, Product, Reason, Description,EmployeeID,
RequestDate, Decision1, ValidComments,ValidationDate FROM EngChange";

        SqlConnection con = new
SqlConnection(ConfigurationManager.ConnectionStrings["ECMS_DatabaseConnectionString"].Con
nectionString);
        SqlCommand cmd = new SqlCommand(selectSQL, con);
        SqlDataReader reader;

        try
        {
            con.Open();
            reader = cmd.ExecuteReader();

            while (reader.Read())
            {
                ListItem newItem = new ListItem();
                newItem.Text = reader["Industry"] + ", " + reader["Product"] + "," +
reader["Reason"] + "," + reader["Description"] + "," + reader["EmployeeID"] + "," +
```

```
reader["RequestDate"] + "," + reader["Decision1"] + "," + reader["ValidComments"] + "," +
reader["ValidationDate"];
                newItem.Value = reader["ChangeName"].ToString();
                DropDownList2.Items.Add(newItem);
            }
            reader.Close();

        }
        catch (Exception err)
        {
            lblMessage.Text = "";
            // lblMessage.Text += err.Message;
        }
    }
    //submitting  the Approval form with new info
    protected void Submit_Click(object sender, EventArgs e)
    {
        //control
        SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["ECMS_DatabaseConnectionString"].Con
nectionString);

        SqlCommand cmd = new SqlCommand("UPDATE EngChange SET Decision2 = @Decision2 ,
DecisionFarctors = @DecisionFarctors, ApprlComments = @ApprlComments, ApprovalDate =
@ApprovalDate WHERE ChangeName = @ChangeName", conn);
        cmd.CommandType = new System.Data.CommandType();


        cmd.Parameters.AddWithValue("@ChangeName", DropDownList2.Text);
        cmd.Parameters.AddWithValue("@Industry", Industry.Text);
        cmd.Parameters.AddWithValue("@Product", Product.Text);
        cmd.Parameters.AddWithValue("@Reason", Reason.Text);
        cmd.Parameters.AddWithValue("@Description", Description.Text);
        cmd.Parameters.AddWithValue("@RequestDate", RequestDate.Text);
        cmd.Parameters.AddWithValue("@EmployeeID", EmployeeID.Text);
        cmd.Parameters.AddWithValue("@Decision1", Decision1.SelectedItem.Text);
        cmd.Parameters.AddWithValue("@ValidComments", Comment1.Text);
        cmd.Parameters.AddWithValue("@ValidationDate", ValidationDate.Text);
        cmd.Parameters.AddWithValue("@Decision2", Decision2.SelectedItem.Text);
        cmd.Parameters.AddWithValue("@DecisionFarctors", DecisionFact.SelectedItem.Text);
        cmd.Parameters.AddWithValue("@ApprlComments", Comment2.Text);
        cmd.Parameters.AddWithValue("@ApprovalDate", ApprovalDate.Text);


        int added = 0;
        try
        {
            conn.Open();
            added = cmd.ExecuteNonQuery();
            lblMessage.Text =  "<html><body><h3 ForeColor=#280000 > Approval successfuly
submitted!</h3></body></Html>";

        }
        catch (Exception err)
        {
            lblMessage.Text = "<html><body><h3 ForeColor=#FF3300> Error occurred during
submitting! </h3></body></Html>";
```

```
            lblMessage.Text += err.Message;
        }
        finally
        {
            conn.Close();
        }

    }
    protected void DropDownList2_SelectedIndexChanged(object sender, EventArgs e)
    {
        //Ddefine ADO.NET Objects
        string selectSQL;

        selectSQL = " SELECT * FROM EngChange";
        selectSQL += " WHERE  ChangeName='" + DropDownList2.SelectedValue + "'";
        SqlConnection con = new
SqlConnection(ConfigurationManager.ConnectionStrings["ECMS_DatabaseConnectionString"].Con
nectionString);
        SqlCommand cmd = new SqlCommand(selectSQL, con);
        //cmd.CommandType = new System.Data.CommandType();
        SqlDataReader reader;

        //Try to open database and read information
        try
        {
            con.Open();
            reader = cmd.ExecuteReader();
            reader.Read();

            //fill the controls

            Industry.SelectedItem.Text = reader["Industry"].ToString();
            Product.SelectedItem.Text = reader["Product"].ToString();
            Reason.SelectedItem.Text = reader["Reason"].ToString();
            Description.Text = reader["Description"].ToString();
            RequestDate.Text = reader["RequestDate"].ToString();
            EmployeeID.Text = reader["EmployeeID"].ToString();
            Decision1.Text = reader["Decision1"].ToString();
            Comment1.Text = reader["ValidComments"].ToString();
            ValidationDate.Text = reader["ValidationDate"].ToString();
            //Email.Text = reader["Email"].ToString();
            reader.Close();
            lblMessage.Text = "";
        }
        catch (Exception err)
        {

        }
        finally
        {
            con.Close();
        }
    }
    protected void Button3_Click(object sender, EventArgs e)
    {
        //Response.Redirect("ApprovalForm2.aspx");
        Response.ClearContent();
```

```
    }
}
```

ValidationForm.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Configuration;
using System.Data.SqlClient;

public partial class EngChangeStatusUpdate : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!this.IsPostBack)
        {
            FillChangeInfo();
        }
        //InitializeCulture();
        if (Session["ecmsRole"] == null)
        {
            Response.Redirect("nologin.aspx");
        }
        else if (Session["ecmsRole"].ToString() != "Manager" &&
Session["ecmsRole"].ToString() != "admin")
        {
            Response.Redirect("notauth.aspx");
        }
    }

    private void FillChangeInfo()
    {
        DropDownList2.Items.Clear();
        string selectSQL = " SELECT Industry, Product, Reason, Description,
ValiadationDate, EmployeeID, Email FROM EngChange";

        SqlConnection con = new
SqlConnection(ConfigurationManager.ConnectionStrings["ECMS_DatabaseConnectionString"].Con
nectionString);
        SqlCommand cmd = new SqlCommand(selectSQL, con);
        SqlDataReader reader;

        try
        {
            con.Open();
            reader = cmd.ExecuteReader();

            while (reader.Read())
            {
                ListItem newItem = new ListItem();
                newItem.Text = reader["Industry"] + ", " + reader["Product"] + "," +
reader["Reason"] + "," + reader["Description"] + " ," + reader["RequestDate"] + "," +
reader["EmployeeID"] ;
                newItem.Value = reader["ChangeName"].ToString();
```

```
                DropDownList2.Items.Add(newItem);
            }
            reader.Close();

        }
        catch (Exception err)
        {
            lblMessage.Text = "";
           // lblMessage.Text += err.Message;
        }
    }

    protected void Submit_Click(object sender, EventArgs e)
    {
        //control
        SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["ECMS_DatabaseConnectionString"].Con
nectionString);

        SqlCommand cmd = new SqlCommand("UPDATE EngChange SET Decision1 = @Decision1,
ValidComments = @ValidComments, ValidationDate = @ValidationDate WHERE ChangeName =
@ChangeName", conn);
        cmd.CommandType = new System.Data.CommandType();


        cmd.Parameters.AddWithValue("@ChangeName", DropDownList2.Text);
        cmd.Parameters.AddWithValue("@Industry", Industry.Text);
        cmd.Parameters.AddWithValue("@Product", Product.Text);
        cmd.Parameters.AddWithValue("@Reason", Reason.Text);
        cmd.Parameters.AddWithValue("@Description", Description.Text);
        cmd.Parameters.AddWithValue("@RequestDate", RequestDate.Text);
        cmd.Parameters.AddWithValue("@EmployeeID", EmployeeID.Text);
        cmd.Parameters.AddWithValue("@Decision1", Decision.SelectedItem.Text);
        cmd.Parameters.AddWithValue("@ValidComments", Comment.Text);
        cmd.Parameters.AddWithValue("@ValidationDate", ValidationDate.Text);




        int added = 0;
        try
        {
            conn.Open();
            added = cmd.ExecuteNonQuery();
            //added.ToString() +
            lblMessage.Text =  "<html><body><h3 ForeColor=#280000 > Validation
successfully submitted!</h3></body></Html>";


        }
        catch (Exception err)
        {
            lblMessage.Text = "<html><body><h3 ForeColor=#280000 > Error occurred during
submitting!</h3></body></Html>";
            lblMessage.Text += err.Message;
        }
        finally
```

```
        {
            conn.Close();
        }

    }
    protected void DropDownList2_SelectedIndexChanged(object sender, EventArgs e)
    {
        //Ddefine ADO.NET Objects
        string selectSQL;

        selectSQL = " SELECT * FROM EngChange";
        selectSQL += " WHERE  ChangeName='" + DropDownList2.SelectedValue + "'";
        SqlConnection con = new
SqlConnection(ConfigurationManager.ConnectionStrings["ECMS_DatabaseConnectionString"].Con
nectionString);
        SqlCommand cmd = new SqlCommand(selectSQL, con);

        SqlDataReader reader;

        //Try to open database and read information
        try
        {
            con.Open();
            reader = cmd.ExecuteReader();
            reader.Read();

            //fill the controls


            Industry.SelectedItem.Text = reader["Industry"].ToString();
            Product.SelectedItem.Text= reader["Product"].ToString();
            Reason.SelectedItem.Text = reader["Reason"].ToString();
            Description.Text= reader["Description"].ToString();
            RequestDate.Text = reader["RequestDate"].ToString();
            EmployeeID.Text = reader["EmployeeID"].ToString();

            reader.Close();
            lblMessage.Text = "";
        }
        catch (Exception err)
        {
            lblMessage.Text = "<html><body><h3 ForeColor=#280000 >Error occurred when
pulling data!</h3></body></Html>";
            //lblMessage += err.Message;
        }
        finally
        {
            con.Close();
        }
    }
    protected void Button3_Click(object sender, EventArgs e)
    {

        Response.ClearContent();
    }
}
```

EngChangeSearch.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class EngChangeSearch : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["ecmsRole"] == null)
        {
            Response.Redirect("nologin.aspx");
        }
    }

}
```

ViewReport.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class ViewReport : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["ecmsRole"] == null)
        {
            Response.Redirect("nologin.aspx");
        }
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        Response.Redirect("CrystalReport41.html");
    }
}
```

35.2 References

1. Gill, P. (2011). Engineering Changes Management Software Tool
   Supply Chain Management at Mid-Sized Enterprises: Efficiently Handling Engineering Changes
   Business Process Requirement Document Ypsilanti: Eastern Michigan University. Retrieved June 16, 2012 from: http://www-personal.umich.edu/~singhpz/Engineering%20Change%20Management%20Software%20Tool.pdf
2. Gorgon, D.; Steheney, T.; Wattas, N.; Yu E. (2005). System Quality Requirements
   Engineering (SQUARE): Case Study on Asset Management System, Phase II. Pittsburgh: Carnegie Mellon University.
3. Introduction to the Personal Software Process, Retrieved June 16, 2012 from, https://docs.google.com/viewer?a=v&q=cache:e_qKDzAdIc8J:www.swenet.org/materials/106/pro2-lecture.ppt+&hl=en&gl=us&pid=bl&srcid=ADGEEShenHndXx2DSO1pfgirAdsE80Xgs-ViNQ-Q8yYhxhYNsJiPSlGUltYgy8jlip4J8y_7mJncqQovMqNyYXQbrTjgHq4TkJ_WKu6CftD4kaaUCbfoyIWI09m6p6oU1xb8__f9m7uA&sig=AHIEtbSc9Rf4A9sn5v7WtHodi_V84KVrMA