

---

# **Software Requirements Specification**

**for**

## **UCM drawing tool**

**Prepared by GROUP:1**

**Team:1**

**March 3, 2099**

# Table of Contents

<b>Table of Contents .....</b>	<b>1</b>
<b>Revision History .....</b>	<b>3</b>
<b>1. Introduction .....</b>	<b>4</b>
1.1 Purpose.....	4
1.2 Document Conventions.....	4
1.3 Intended Audience and Reading Suggestions .....	4
1.4 Project Scope.....	5
1.5 References.....	5
<b>2. Overall Description .....</b>	<b>5</b>
2.1 Product Perspective.....	5
2.2 Product Features.....	6
2.3 User Classes and Characteristics.....	7
2.4 Operating Environment.....	8
2.5 Design and Implementation Constraints .....	8
2.6 User Documentation .....	8
2.7 Assumptions and Dependencies.....	9
2.8 Budget.....	9
<b>3. System Features .....</b>	<b>11</b>
3.1 Overall Project Features.....	11
3.2 UCM drawing tool and store UCM data subproject. ....	13
3.2.1 UCM drawing tool and store UCM data domain model .....	13
3.2.2 UCM drawing tool and store UCM data Use Cases Diagram. ....	14
3.3 Move Line / Path.....	15
3.3.1 Description and Priority .....	15
3.3.2 Stimulus/Response Sequences .....	15
3.3.3 Functional Requirements .....	17
3.4 Move Component.....	18
3.4.1 Description and Priority .....	18
3.4.2 Stimulus/Response Sequences .....	18
3.4.3 Functional Requirements .....	18
3.5 Edit Label.....	18
3.5.1 Description and Priority .....	18
3.5.2 Stimulus/Response Sequences .....	19
3.5.3 Functional Requirements .....	19
3.6 Edit Diagram.....	19
3.6.1 Description and Priority .....	19
3.6.2 Stimulus/Response Sequences .....	19
3.6.3 Functional Requirements .....	19
3.7 Make Diagram.....	20
3.7.1 Description and Priority .....	20
3.7.2 Stimulus/Response Sequences .....	20
3.7.3 Functional Requirements .....	20
3.8 Make Line .....	20

3.8.1	Description and Priority .....	20
3.8.2	Stimulus/Response Sequences .....	20
3.8.3	Functional Requirements .....	22
3.9	Make Component.....	22
3.9.1	Description and Priority .....	22
3.9.2	Stimulus/Response Sequences .....	23
3.9.3	Functional Requirements .....	24
3.10	Make Junction .....	24
3.10.1	Description and Priority .....	24
3.10.2	Stimulus/Response Sequences .....	25
3.10.3	Functional Requirements .....	25
3.11	Make Label .....	25
3.11.1	Description and Priority .....	25
3.11.2	Stimulus/Response Sequences .....	26
3.11.3	Functional Requirements .....	28
3.12	Make Responsibility .....	28
3.12.1	Description and Priority .....	28
3.12.2	Stimulus/Response Sequences .....	28
3.12.3	Functional Requirements .....	28
3.13	Make Point.....	29
3.13.1	Description and Priority .....	29
3.13.2	Stimulus/Response Sequences .....	29
3.13.3	Functional Requirements .....	29
3.14	File Operations .....	29
3.14.1	Description and Priority .....	29
3.14.2	Stimulus/Response Sequences .....	30
3.14.3	Functional Requirements .....	30
<b>4.</b>	<b>External Interface Requirements .....</b>	<b>30</b>
4.1	User Interfaces .....	30
4.2	Hardware Interfaces .....	31
4.3	Software Interfaces .....	31
4.4	Communications Interfaces.....	31
<b>5.</b>	<b>Other Non-Functional Requirements .....</b>	<b>31</b>
5.1	Performance Requirements .....	31
5.2	Safety Requirements .....	31
5.3	Security Requirements .....	31
5.4	Software Quality Attributes .....	31
<b>6.</b>	<b>Other Requirements .....</b>	<b>32</b>
<b>7.</b>	<b>Appendix A: Glossary.....</b>	<b>32</b>
<b>8.</b>	<b>Appendix B: Issues Lists .....</b>	<b>32</b>

## Revision History

Student	Use Case	Sections
Your name and Student-id here	Edit Line	2.8 and 3
Your name and Student-id here	Make Line	5 and 6
Your name and Student-id here	Make Label	1 and 2.1 – 2.7
Your name and Student-id here	Make Component	4

## 1. Introduction

The UCM drawing tool is a software program that allows a user to create and edit UCM's or USE CASE Maps a notation for helping a person visualize, think about, and explain the overall behaviour of a complex system [1]. Our project group will be focused on the implementation of the UCM drawing tool using the Java programming language as well as storing the UCM data. This document follows Dr. Rilling SRS [2] slides to specify the system requirements and describe the system design. Based on the course of the Computer Science Department at Concordia University. This document follows the SRS template downloaded from [www.processimpact.com](http://www.processimpact.com) and will follow the object oriented design methodology.

### 1.1 Purpose

This software product is identified as “UCM drawing tool” Version 1.0. This SRS will describe the part of the system that is concerned with the creation, editing, saving and deleting of UCM files as well as describe the user login. The system and documentation are to be designed in terms of usability and user friendliness.

### 1.2 Document Conventions

This document uses the IEEE standard Times New Roman size 12 fonts. Bold/Underline is used to convey important terms. Every requirement statement will have its own priority.

### 1.3 Intended Audience and Reading Suggestions

This document is intended for all audience types, from the technically savvy person to the common everyday user. It is organized into 6 sections: 1) Introduction 2) Overall Description 3) System Features 4) External Interface Requirements 5) Other Nonfunctional Requirements 6) Other Requirements. **Developers** may want to read sections 1.1 to 1.5, 2.1, 2.4, 2.5, 2.7, 2.8, 3.2, 3.3, 4.1 to 4.4, and 5.1 to 5.4. **Project Managers** may want to read sections 1.1 to 1.5, 2.1, 2.4, 2.5, 2.8, 3.1 to 3.4, and 5.1 to 5.4. **Marketing staff** will be interested in sections 1.1 to 1.5, 2.2, 2.6, and Section 4.1. **Everyday Users** will want to read sections 1.1 to 1.5, 2.2, 2.4, 2.6, 4.1, 4.4, and 5.2 to 5.4. **Testers** will want to read sections 1.1 to 1.5, 2.2, 2.4, 2.6, 2.7, 4.1 to 4.4 and 5.1 to 5.4. **Documentation writers** may be interested in sections 1.1 to 1.5, 2.1 to 2.7, 3.1, 3.3, 4.1 to 4.4, 5.2 and 5.3.

## 1.4 Project Scope

The UCM drawing tool is a software engineering tool intended to be useful for requirements specification, design, testing, maintenance, adaptation, and evolution [1]. The company outsourcing this project simply wants an effective, user friendly product of superior quality. The company wishes that this software product competes with similar products currently available. In order to give it an edge, the company would like the software to have more features than its competitors. This is the company's first attempt at this kind of software. This SRS will be concerned with user login, create file, delete file, save file, open file, close file, Make Diagram, Edit Diagram which includes Lines, Points, Components, AND/OR join forks, Labels and Responsibilities.

The deliverable products are:

UCM drawing tool program: implemented in Java

UCM drawing tool document: This document will take the reader through the software life cycle: Requirements analysis + specification, Design, Implementation, Testing and maintenance.

## 1.5 References

[1]

<http://jucmnav.softwareengineering.ca/ucm/pub/UCM/VirLibTransactionsSE98/ucmUpdate.pdf>

[2] Juergen Rilling, **Understanding Requirements** course material, Concordia University, Department of Computer Science,

## 2. Overall Description

### 2.1 Product Perspective

The UCM drawing tool being developed for **MyUCM2011** Software Engineering Company is a new self-contained product. It contains the following features:

Line: Allows user to create and modify Lines

Component: Allows user to create and modify Components

Junction: Creates a new OR/AND join or fork

Responsibilities: Creates a new Responsibility

Point: Creates a new Point

Label: Allows user to create and modify a Label for a feature

Diagram: Creates and initializes the diagram of a file

File: Options are New, Delete, Save, Open, Close

## 2.2 Product Features

### Line:

1. Create a new Line
2. Edit a created Line
3. Move a created Line

### Junction:

1. Create AND fork
2. Create OR fork
3. Create AND join
4. Create OR join

### Responsibilities:

1. Create a new Responsibility

### Point:

1. Create a new Point (beginPoint, endPoint)

### Component:

1. Create a new Component
2. Move a Component

### Label:

1. Create a new Label
2. User enters Label name
3. Edit previously created Label

### Diagram:

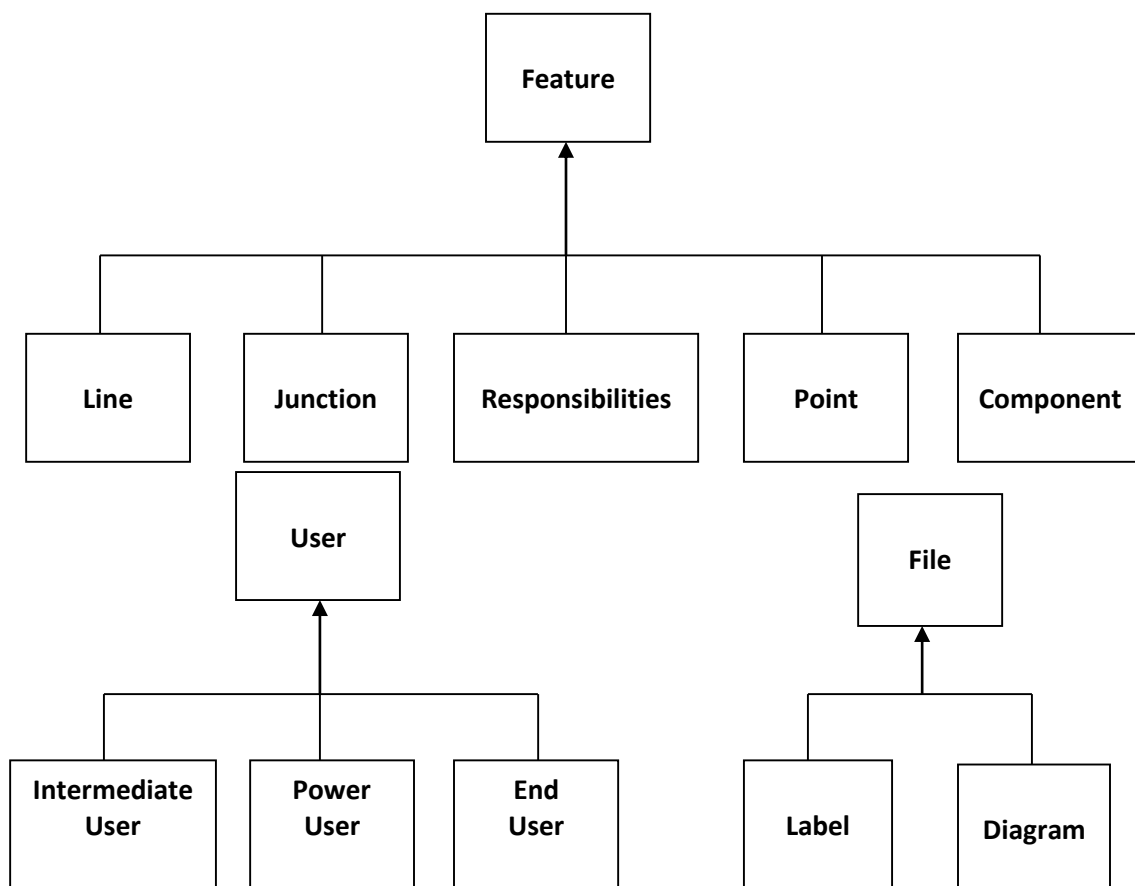
1. Create a new Diagram
2. User enters Diagram name
3. Edit previously created Diagram

### File:

1. Create a New File
2. Delete a File
3. Save a File
4. Open a previously saved File

## 5. Close a File

**Class Diagram**



## 2.3 User Classes and Characteristics



The users of the UCM drawing tool will vary from Software Engineering students to Software Engineering professionals. We will categorize the users of the UCM drawing tool as **Power Users**, **Intermediate Users** and **End Users**. The **Power User** is highly educated, experienced and has a high level of technical expertise. He/She is most likely a professional Software Engineer possibly a leader of a group of Software Engineers working on a project. He/She will have the highest security/privilege level and will use the product more than any other user. Therefore, the Software Engineer is the favoured user class of the UCM drawing tool. The **Intermediate User** is educated, has some technical expertise and some experience. He/She is a working Software Engineering professional or Graduate Student, and will have some privileged access. He/She will use the product some of the time. The **End User** has low or no technical expertise and low privileges. Possibly a student or customer, he/she will use the product very minimally and just for reference.

## 2.4 Operating Environment

The operating system will be Microsoft Windows 7 running on the x86 hardware platform. Eclipse SDK Version 3.2.1 will be used. Other software components include Eclipse Java Development Tools, Eclipse Plug-In development environment, etc.

## 2.5 Design and Implementation Constraints

The programming language used to implement the UCM drawing tool will be Java. The user will have limited access to the 'Make Diagram' tool which includes: Make Line, Make Point, Make Component, Make Junction, Make Label, and Make Responsibility. The 'New File' and 'Delete File' will also have limited access. There will also be some user restrictions on the 'Edit Diagram' and 'Save File' tools. Future upgrades and maintenance of the software will be handled by **MyUCM2011** Software Engineering Company.

## 2.6 User Documentation

The user documentation standard we will use is **List or Reference**. The commands will be listed alphabetically and indexed. This standard will appeal to the advanced user. [4]

[4] [http://en.wikipedia.org/wiki/Software\\_documentation](http://en.wikipedia.org/wiki/Software_documentation)

## 2.7 Assumptions and Dependencies

The Feature class is composed of 5 sub-classes. The User class is composed of 3 sub-classes. The object oriented method of inheritance is assumed here.

Other assumptions and dependencies:

- A diagram contains features and labels mark features.
- The development requires the Microsoft Windows operating system.
- There will be 3 user types: Power, Intermediate and End.
- There will be limited access to the system.

At this point, it is not clear whether or not we will use software components from another project.

## 2.8 Budget

We estimate our budget for this part of the project will be **\$16,705.00**. We used two techniques (COCOMO and Bottom up Estimations) to calculate the total cost we'll incur in creating the UCM drawing tool and data store. Both estimations derive in basically the same result and we picked the COCOMO estimation as our budget. The explanation on how we derived each of the estimations follows.

2.8.1. Our first estimation is based on the Bottom up Estimation method. The calculated estimated cost is **\$16,666.67**. The number of work days needed for the design, implementation, testing and release of the project with our sub-team of 4 people is the following:

1. *Sub-project requirements, Specification, domain analysis, Software architecture*: 8 days
2. *Sub-project implementation and testing*: 15 days

Our domain model contains 12 object classes we think represent the domain problem of the UCM drawing tool and data store subproject. We will use several existing libraries that don't need additional coding and up to 4 supporting or helper and testing classes. From our past experience we know we will need an average of 100 new lines of code per class. So we expect to have 1600 total lines of code. Additionally, our team of 4 people has an average programming rate of 27 productive lines of code per an 8 hour work day each, therefore we will need  $1600/(4*27) = 15$  days to finish the product.

- User Class and inherited Power, end and Intermediate users: 2 days of work
- File Class: 1 days of work
- Feature, Label, Junction and Point Classes: 1 day

- Line Class: 2 days
- Responsibilities Class: 1 day
- Component Class: 2 day
- Helper classes: 2 days
- Testing: 4 days

3. *Sub-Product release*: 2 days.

We think 25 days will be enough to finish this part of the whole project. Therefore the total bottom up estimated cost of our system will be (based on a monthly pay rate of \$5,000.00 and pro-rated to hour total days of work):

**Bottom up Estimated Cost** =  $\$4,166.67 * 4 = \$16,666.67$

2.8.2. The second estimation is based on the Boehm's Constructive Cost Model, COCOMO. The Total Estimated Cost based on this calculation is **\$16,705.00**.

Based on the specifications and parameters of this model, we believe our project is of type Organic as it is small in size requires a small amount of innovation and does not have very tight deadlines. We have also decided that our adjustment factor for this project will be **0.687789**. The relevant criteria/factors in the calculation of the adjustment factor were based on our experience in developing this type of software, the user requirements and the software specification:

- Product Complexity (Low): 0.85
- Required Software Reliability (High) : 1.15
- Applications Experience (High) : 0.91
- Programmer capability (High): 0.86
- Programming Language Experience (High): 0.95
- Use of software Tools (High) : 0.91
- Required Development Schedule (High): 1.04

We also have estimated that the total lines of code (KLOC, measured in the thousands) needed to complete the project will be around 1600 (see previous point). We came up with this figure by the multiplication between the total classes and additional supporting classes our system domain model has, and an average of 27 lines of code per day for each person in the group (this last figure is based on our past experience). We also estimate that the project will be completed in a month.

Therefore, the total estimated cost for this part of the project is the following:

Effort per person per month =  $2.4 * 1.6^{1.5} * 0.687789 = 3.341$  **person/months**

And the total estimated cost is:

**Total Estimated Cost:**  $3.341 * \$5,000.00 = \$16,705.00$

(Based on a monthly salary rate of \$5,000.00)

Complete team's Cost estimation Analysis:

In order to find the total cost estimation for the team, which is composed from two groups, Group A and Group B, we will sum up the total amount of the estimates.

The estimated budget that group A has calculated: \$16,705.00

The estimated budget that group B has calculated: \$11,080.00

Therefore, the total cost for the entire team is estimated to be:

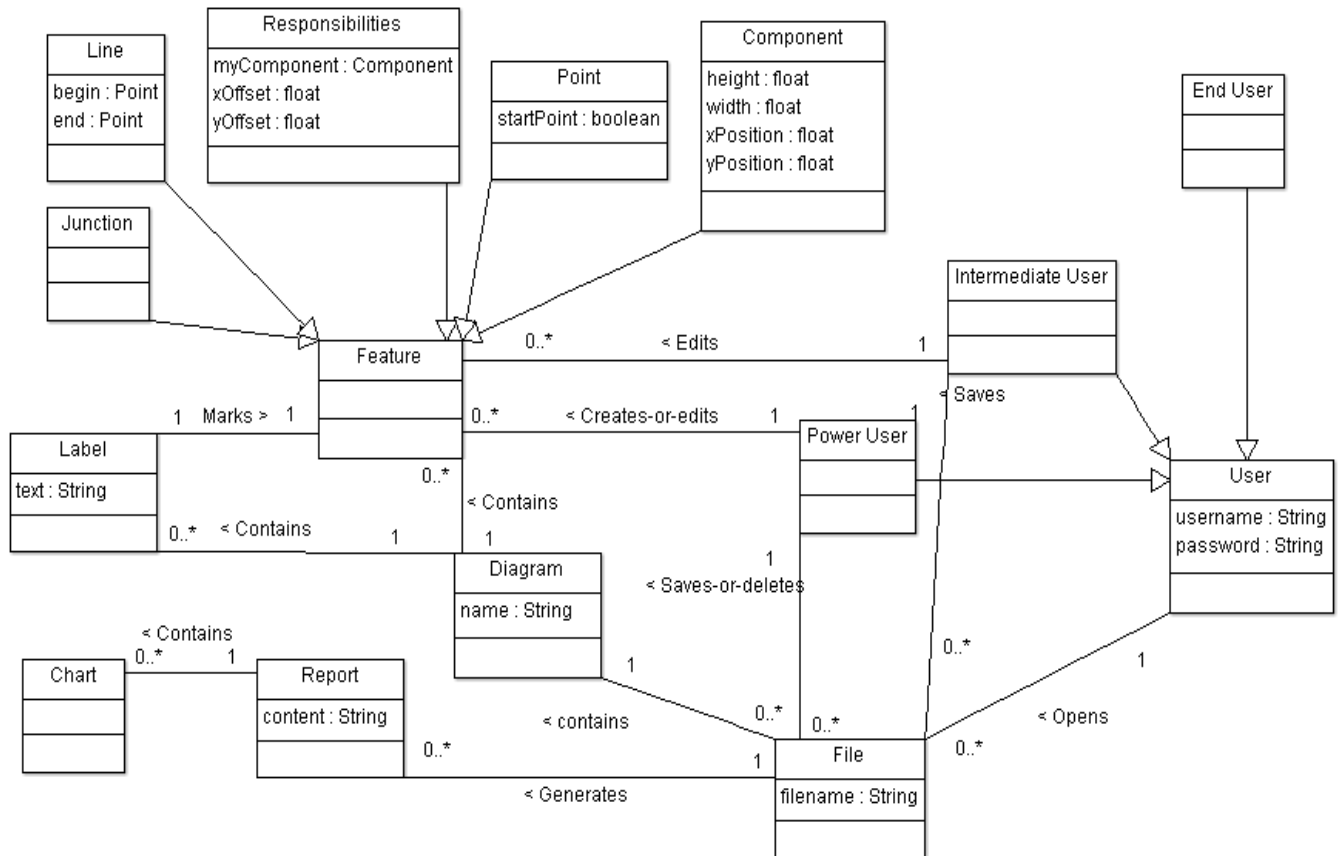
***Total Cost for the whole Team= \$27,785.00***

### **3. System Features**

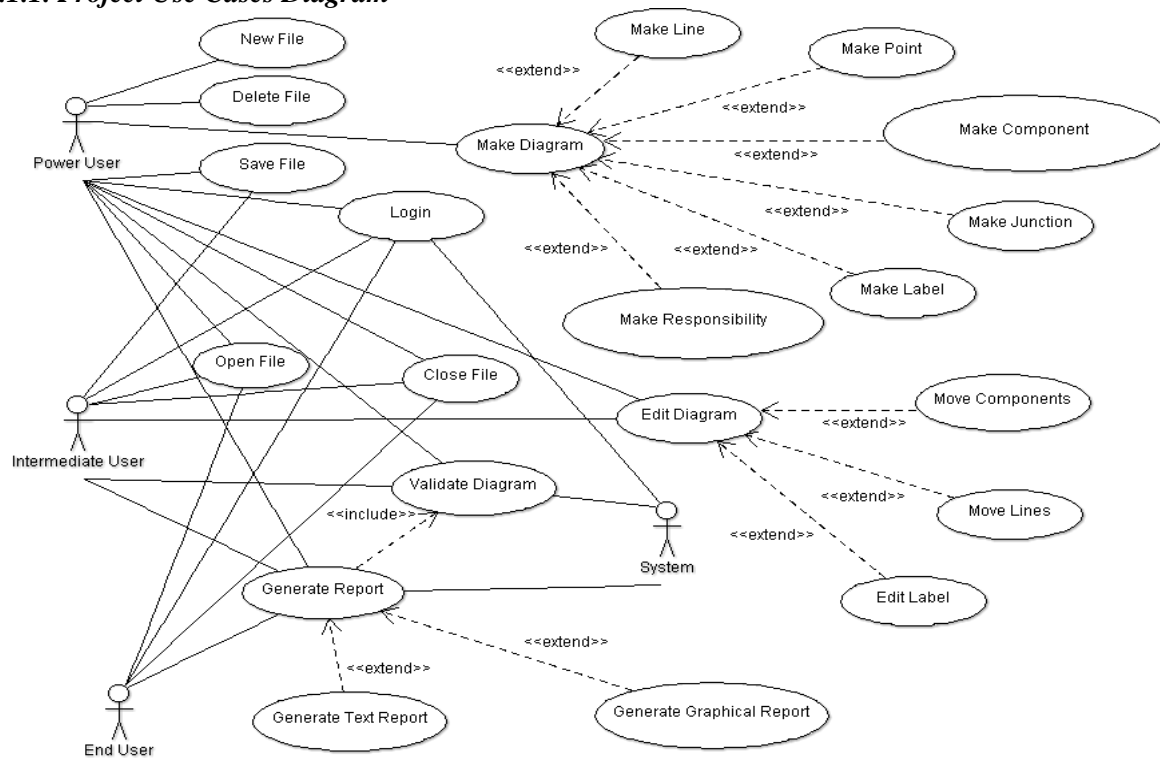
The project aims to provide a simple UCM tool using an easily accessible GUI. Files are saved as and parsed from XML files, a simple, open file format. The system also includes a validator to ensure that created UCMs are well formed and unambiguous and allows for the gathering and presenting of statistical data regarding the UCM through both textual and graphical reports.

#### **3.1 Overall Project Features**

##### ***3.1.1. Project Domain Model***



### 3.1.1. Project Use Cases Diagram

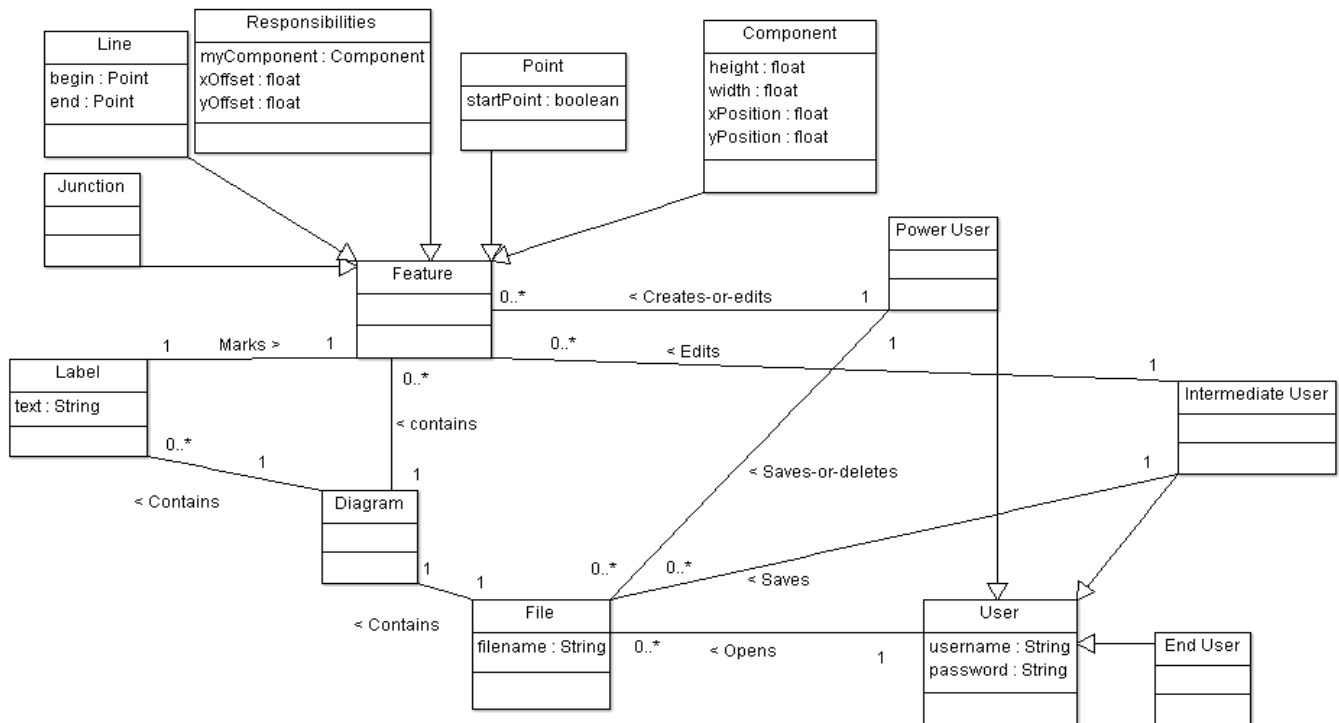


## 3.2 UCM drawing tool and store UCM data subproject.

This part of the project will design and implement the Use case Maps drawing tool and store the UCM features in a XML file.

### 3.2.1 UCM drawing tool and store UCM data domain model

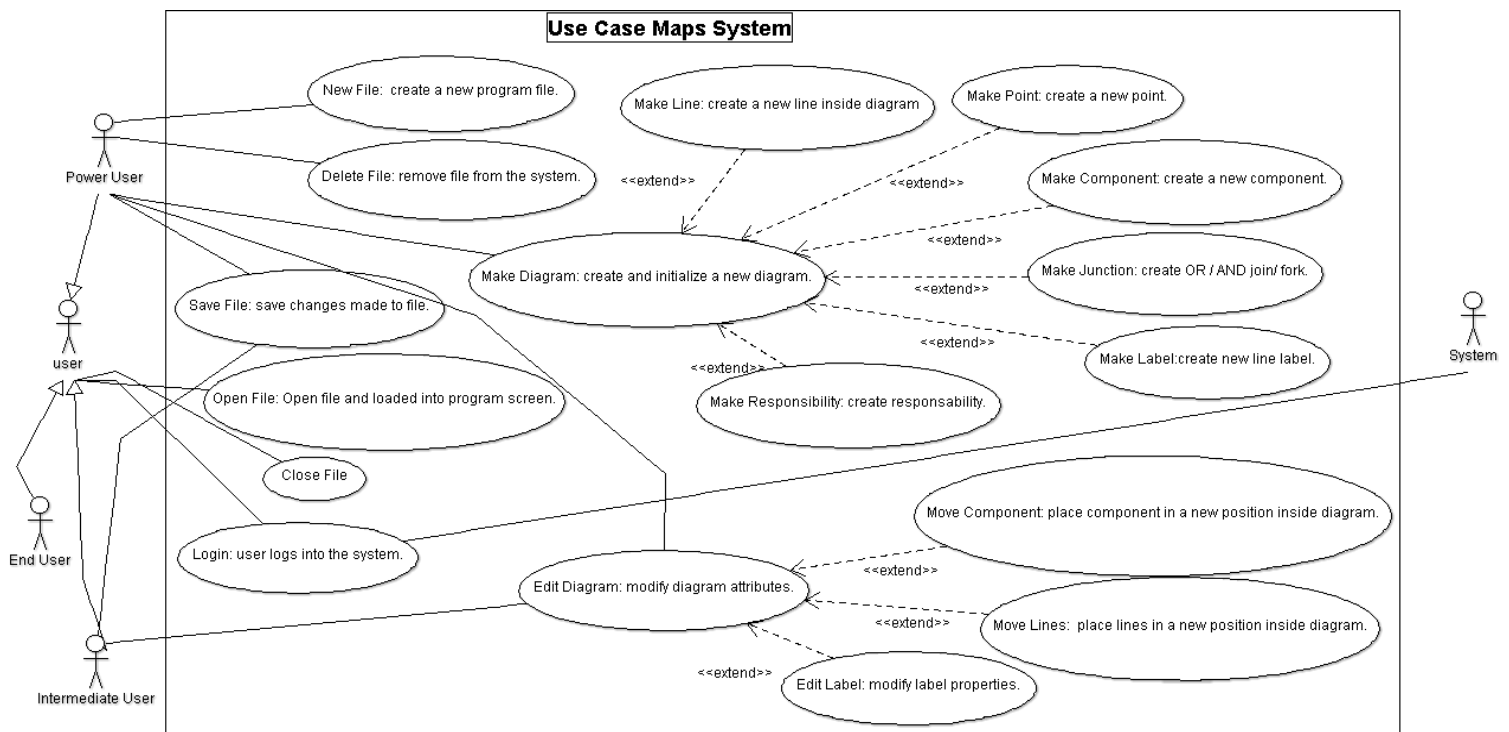
We have decided that the domain problem for our sub-project can be represented by the following objects and relations.



**Figure: UCM drawing tool and store UCM data domain model**

### 3.2.2 UCM drawing tool and store UCM data Use Cases Diagram.

The Use cases diagram for our subproject is the following:



**Figure: UCM drawing tool and store UCM use cases Diagram with brief descriptions.**

### 3.3 Move Line / Path

#### 3.3.1 Description and Priority

This feature places the selected Line object in a new position inside diagram. This use case is of high priority as it deals with the movement of one of the main elements of UCM system.

- Benefit: 9
- Penalty: 3
- Cost: 3
- Risk: 5

#### 3.3.2 Stimulus/Response Sequences

- User requests to move a Line.
- System presents edition context to pass attributes values.
- User enters attribute values.
- System modifies Line attributes with provided values, places Line in diagram in position indicated by attributes and shows Line to the screen.

As a list of user actions and system responses for this feature we present the system sequence diagram, sequence diagram and collaboration diagram for the success scenario of this use case:

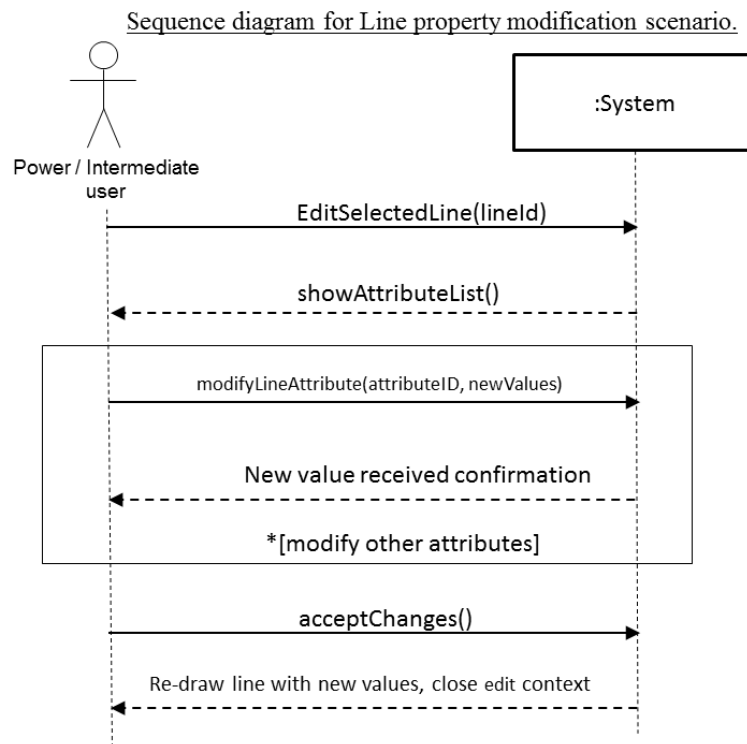


Figure: System Sequence diagram for success scenario of Move Line use case.



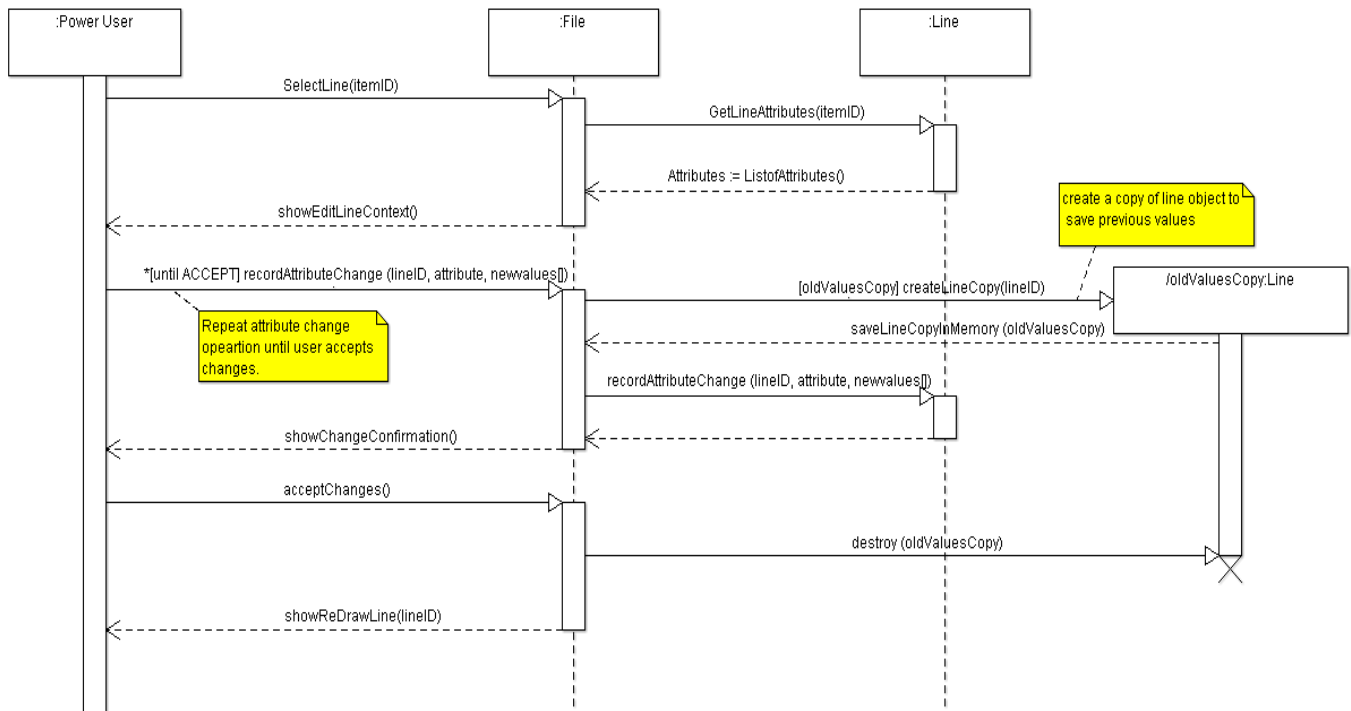


Figure: Sequence diagram for success scenario of Move Line use case.

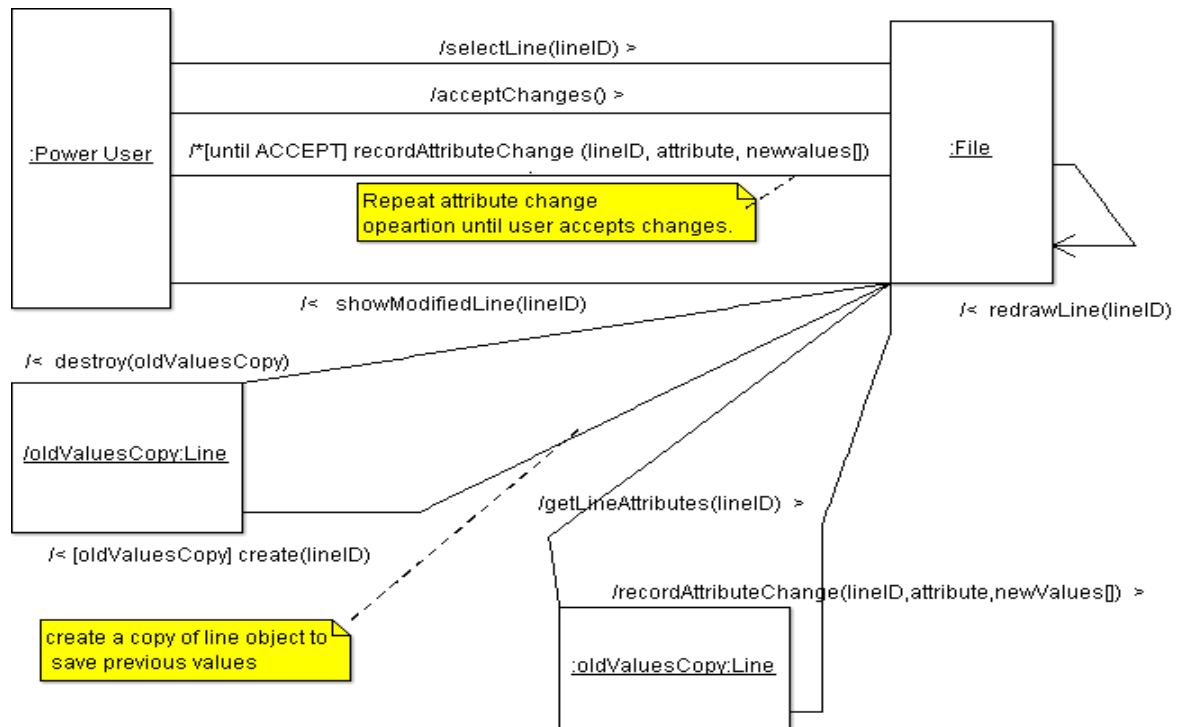


Figure: Collaboration diagram for success scenario of Move Line use case.

### 3.3.3 Functional Requirements

- REQ-1: User must be logged in as either a Power or Intermediate user, show message to user if this is not the case.
- REQ-2: If user does not provide values for position attributes, load default values with position (0, 0).
- REQ-3: Every Line must have a beginning and end Point. Notify user with an error message.

We present the fully dressed description of the Move Line feature use case to explain the functional requirements.

#### Use Case: Move Line /Path

Primary Actor: Intermediate or Power User

Stakeholders and Interests: - Intermediate or Power User: Wants no errors modifying lines and wants quick changes.

Preconditions: Users is logged into the system as a Power or Intermediate user. At least a line has been drawn in diagram. User has identified the path/line he wants to edit.

Post conditions: Line/path attributes are modified and line is moved to desired position. Line is ready to be saved to file. GUI shows changes to the screen.

#### Main success scenario:

- User selects line/path to move and opens the Edition context to start process.
- List of attributes for selected line is shown to the screen.
- User scans through the list of attributes and selects the one she wants to modify.
- User modifies current attribute values by deleting the current values and typing the new ones, sends new values to the system. System confirms changes. User repeats steps 3 and 4 to modify as many line attributes as she wants.
- After all modifications are done, user accepts changes and notifies the system.
- System records and shows the modifications to the screen by using the new values of each attribute to re-draw the line into the new position and show it to the screen.
- System closes Edit context.

#### Alternative scenario:

If user enters and incorrect value for a given attribute system notifies user changes cannot be recorded until valid values are provided.

If errors occur when system is trying to record modifications or is re-drawing the line, the previous attributes values are restored and modifications are cancelled. User is notified of errors.

If user cancels modifications, restore previous values.

Special Requirements: A Line must always have a beginning and end Point.

Open Issues: define the REQUIRED line attributes.

## 3.4 Move Component

### 3.4.1 Description and Priority

This feature places the selected Component object in a new position inside diagram. This use case is of high priority as it deals with the movement of one of the main elements of UCM system.

- Benefit: 9
- Penalty: 5
- Cost: 3
- Risk: 1

### 3.4.2 Stimulus/Response Sequences

- User requests to move a Component.
- System presents edition context to pass attributes values.
- User enters attribute values.
- System modifies Component attributes with provided values, places Component in diagram in position indicated by attributes and shows Component to the screen.

### 3.4.3 Functional Requirements

- REQ-1: User must be logged in as a Power or Intermediate user, show message to user if this is not the case.
- REQ-2: Components must have a Rectangular like form. If user does not provide values for position attributes, load default values with position (0, 0) and height and width of 10.

## 3.5 Edit Label

### 3.5.1 Description and Priority

Feature modifies the label attributes of the selected Feature (Line, Component, etc.). Attributes of the label include: Text, format, font and weight. This use case is of low priority as features in our system can exist without labels.

- Benefit: 5
- Penalty: 1
- Cost: 1
- Risk: 1

### 3.5.2 Stimulus/Response Sequences

- User selects a feature and requests the system to edit its label.
- System shows label attributes to be modified. User modifies attributes and notifies system
- System records label attributes and re-draws label inside diagram.

### 3.5.3 Functional Requirements

- REQ-1: User must be logged in as either a Power or Intermediate user, show message to user if this is not the case.
- REQ-2: Label must be part of a Feature.
- REQ-3: Label Attribute “Text” must have a length of 25 characters. Prompt user that she needs to change the text when editing if this is not the case.

## 3.6 Edit Diagram

### 3.6.1 Description and Priority

This feature modifies the attributes of the Diagram. This use case is of low priority as it only modifies the attributes of the diagram and delegates the modification of its elements to the other operations of the system.

- Benefit: 1
- Penalty: 1
- Cost: 1
- Risk: 1

### 3.6.2 Stimulus/Response Sequences

- User selects diagram and asks to modify its attributes
- System shows diagram attributes (name, etc.) to be modified. User modifies attributes and notifies system
- System records modification and shows it to the screen.

### 3.6.3 Functional Requirements

- REQ-1: User must be logged in as either a Power or Intermediate user, show message to user if this is not the case.
- REQ-2: Diagram must have a name, a notification to user is shown when she tries to leave this attribute blank.

## 3.7 Make Diagram

### 3.7.1 Description and Priority

This feature creates and initializes the sole diagram of a given file in our drawing system. This use case is of very high priority as the diagram is the canvas where the rest of the Features (Line, etc.) will be drawn.

- Benefit: 9
- Penalty: 8
- Cost: 8
- Risk: 2

### 3.7.2 Stimulus/Response Sequences

- User enters Diagram name and asks to create diagram.
- System verifies no diagram has already been created, creates diagram, saves file and shows it to the screen.

### 3.7.3 Functional Requirements

- REQ-1: User must be logged in as either a Power user, show message to user if this is not the case.
- REQ-2: File must be created and opened.
- REQ-3: Diagram must have a name, a notification to user is shown when she tries to leave this attribute blank.

## 3.8 Make Line

### 3.8.1 Description and Priority

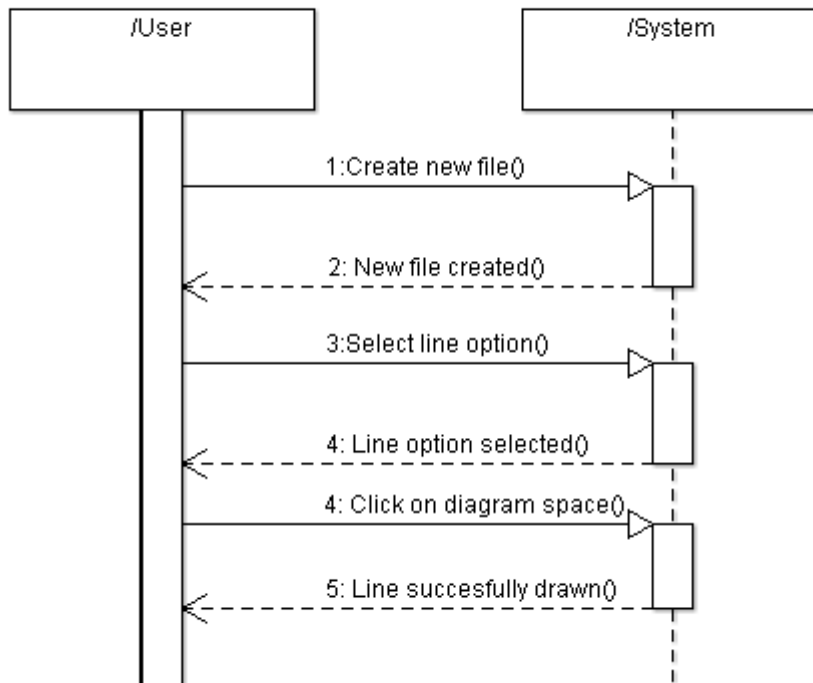
This feature creates a new Line inside a diagram. This use case is of high priority as a Line shows the path of any system represented by the use case.

- Benefit: 9
- Penalty: 3
- Cost: 3
- Risk: 1

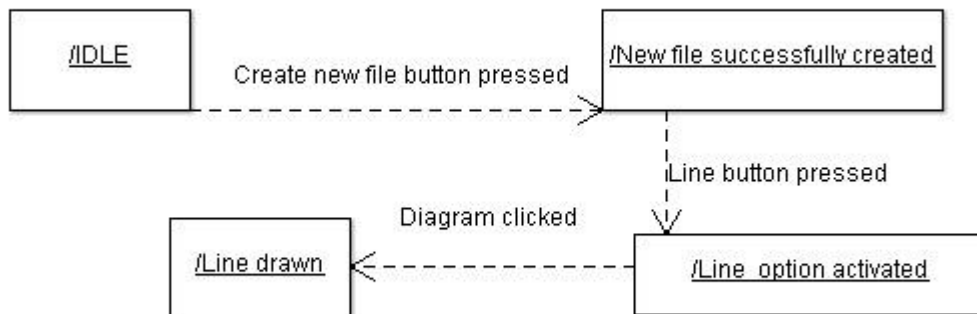
### 3.8.2 Stimulus/Response Sequences

- User requests to create a line.
- System presents creation context to pass attributes values.
- User enters attribute values.

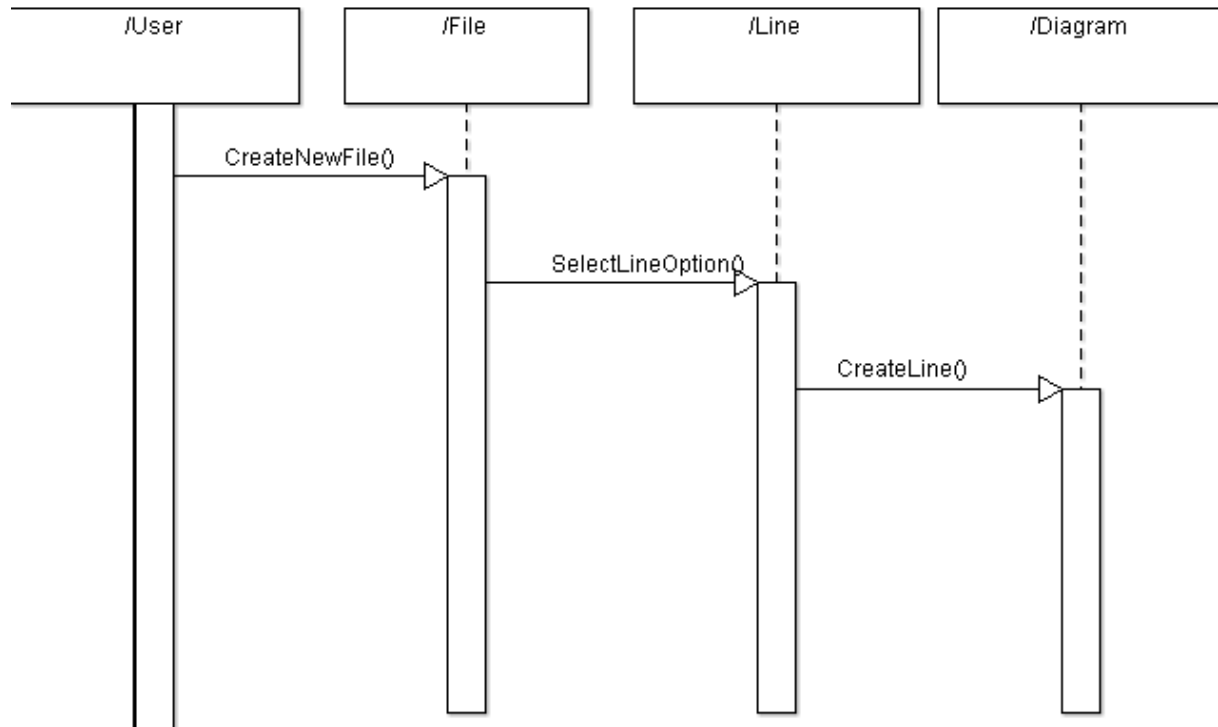
- System creates line, loads attributes with default values, places line in diagram in position indicated by attributes, creates two end Point objects (a beginning and an end) and shows line to the screen.



System Sequence Diagram



Collaboration Diagram



Sequence Diagram

### 3.8.3 Functional Requirements

- REQ-1: User must be logged in as a Power user, show message to user if this is not the case.
- REQ-2: The file diagram must be already created. Show message to user notifying about she needs to create diagram before creating a new line.
- REQ-3: If user does not provide values for position attributes, load default values with position (0, 0).
- REQ-4: Every Line must have a beginning and end Point.

## 3.9 Make Component

### 3.9.1 Description and Priority

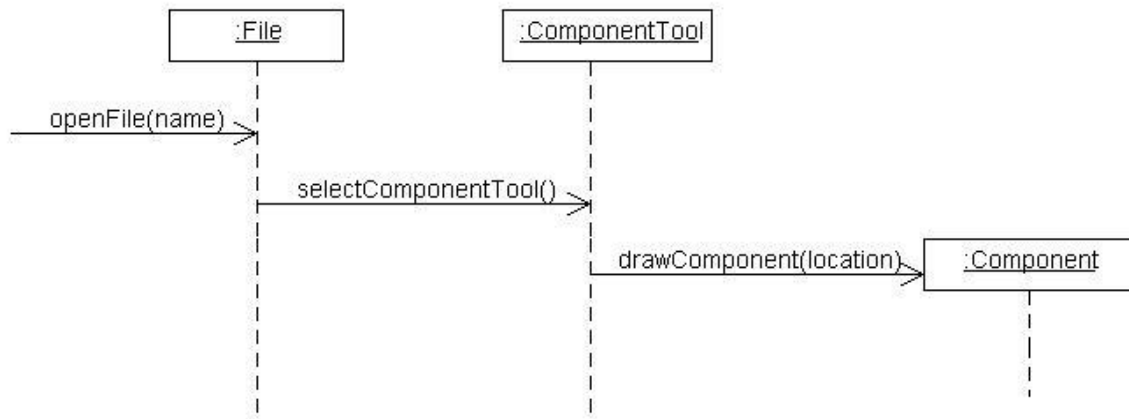
This feature creates a new Component inside the diagram. This use case is of high priority as a Component represents a system or an object in the structure represented by the use case.

- Benefit: 9
- Penalty: 5
- Cost: 5
- Risk: 2

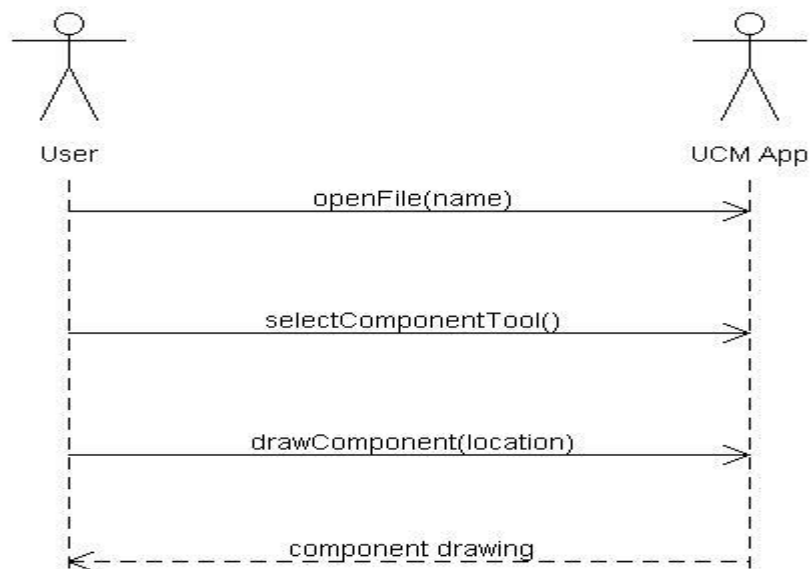
### 3.9.2 Stimulus/Response Sequences

- User requests to create a Component.
- System presents creation context to pass attributes values.
- User enters attribute values.
- System creates Component, loads attributes with default values, places Component in diagram in position indicated by attributes and shows Component to the screen.

#### Sequence Diagram:



#### System Sequence Diagram:



#### Collaboration Diagram:





### 3.9.3 Functional Requirements

- REQ-1: User must be logged in as a Power user, show message to user if this is not the case.
- REQ-2: The file diagram must be already created. Show message to user notifying about she needs to create diagram before creating a new component.
- REQ-3: Components must have a Rectangular like form. If user does not provide values for position attributes, load default values with position (0, 0) and height and width of 10.

**Use Case Make Component:** The user wants to draw a component on the canvas.

Actor(s): User

Pre-conditions:

- The UCM Drawing Tool application is open and running under the JVM.
- Drawing Component tool is selected.
- File is open

Main Scenario:

1. The user selects the component tool button from the toolbar.
2. Using the drawing component tool the user draws the component element onto the canvas.

Alternate Scenario 1:

1. The user attempts to draw outside of the canvas and the drawing is not executed.
2. A window dialog appears with the respective error message.

Post-conditions:

- The component has been drawn onto the canvas.

## 3.10 Make Junction

### 3.10.1 Description and Priority

This feature creates a new “OR / AND” join or fork inside the diagram. This use case is of medium priority as the Junction is not an essential or necessary element of the system represented by the use case.

- Benefit: 7
- Penalty: 5
- Cost: 3
- Risk: 3

### 3.10.2 Stimulus/Response Sequences

- User requests to create a Junction.
- System presents creation context to pass attributes values.
- User enters attribute values.
- System creates Junction, loads attributes with default values, places Junction in diagram in position indicated by attributes and shows Junction to the screen.

### 3.10.3 Functional Requirements

- REQ-1: User must be logged in as a Power user, show message to user if this is not the case.
- REQ-2: The file diagram must be already created. Show message to user notifying about she needs to create diagram before creating a new component.

## 3.11 Make Label

### 3.11.1 Description and Priority

This feature creates a new Label for a given Feature (Component, Line, etc.) inside diagram. This use case is of low priority as any Feature in the system can exist without a label.

- Benefit: 5
- Penalty: 1
- Cost: 1
- Risk: 1

Primary Actor: Power User

Preconditions: User is logged in to the system as a Power User. At least a line or a component has already been created. User must have selected line or component to request to make Label for it.

Post conditions: The System creates the Label and attaches it to the line or the component. The system then places the Label in the diagram and shows results to the screen.

Main Success Scenario:

- 1. User selects Line/Component and makes create Label request.
- 2. System shows user options.
- 3. User enters values for the Label.

- 4. System confirms values and creates the Label.
- 5. System attaches Label to Line/Component.
- 6. System places Label in diagram position indicated by the user.
- 7. System shows Label to the screen.

### 3.11.2 Stimulus/Response Sequences

Figure: System Sequence Diagram for success scenario of Make Label use case

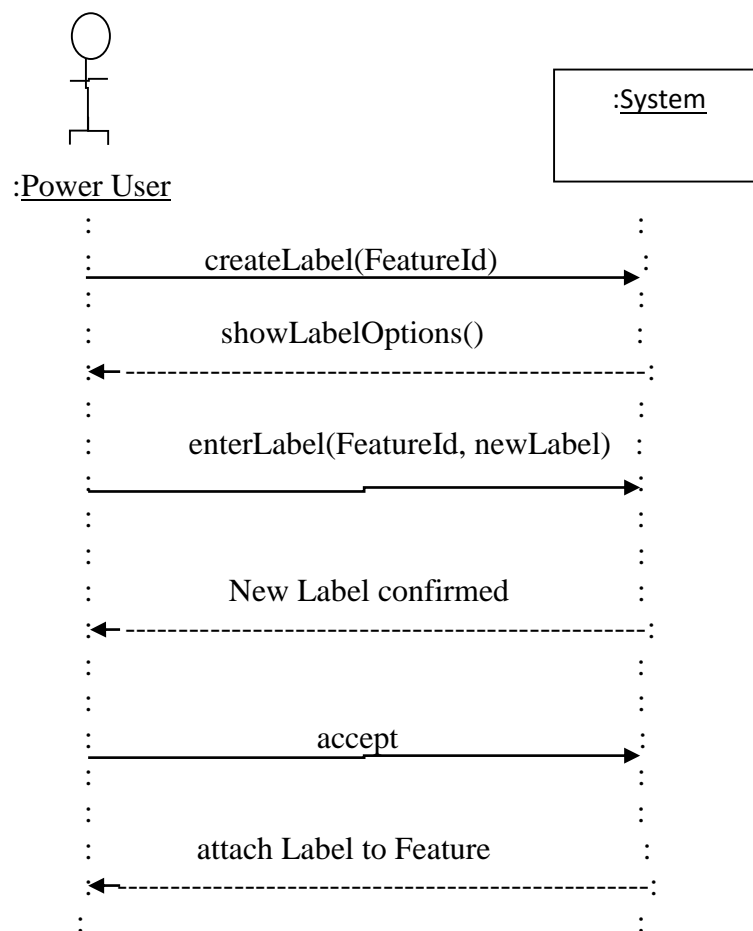
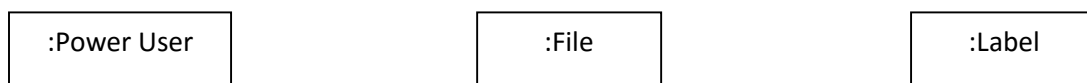


Figure: Sequence Diagram for success scenario of Make Label use case



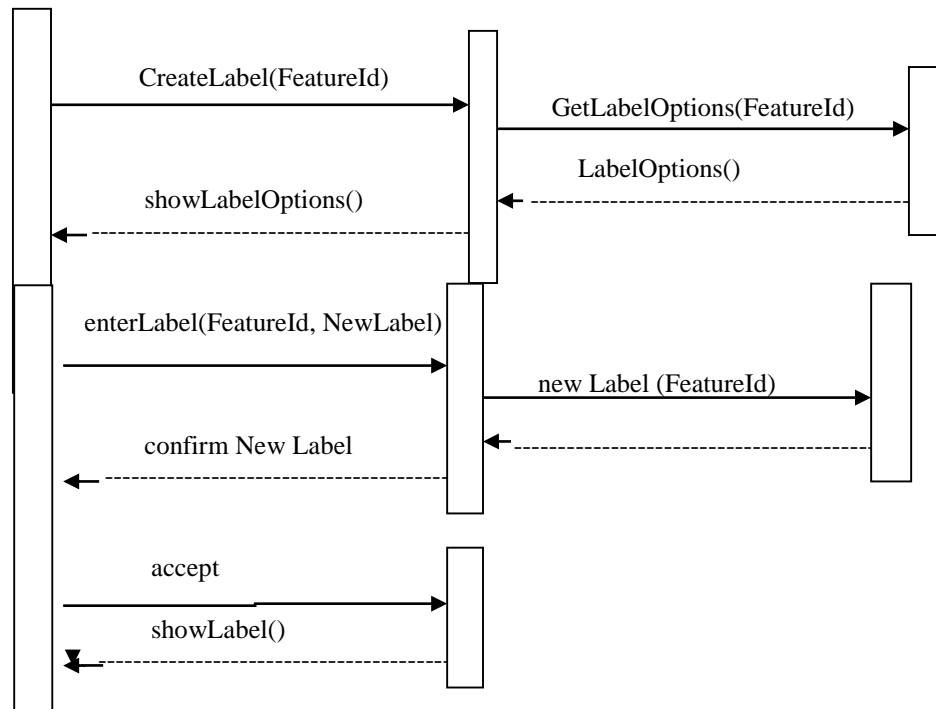
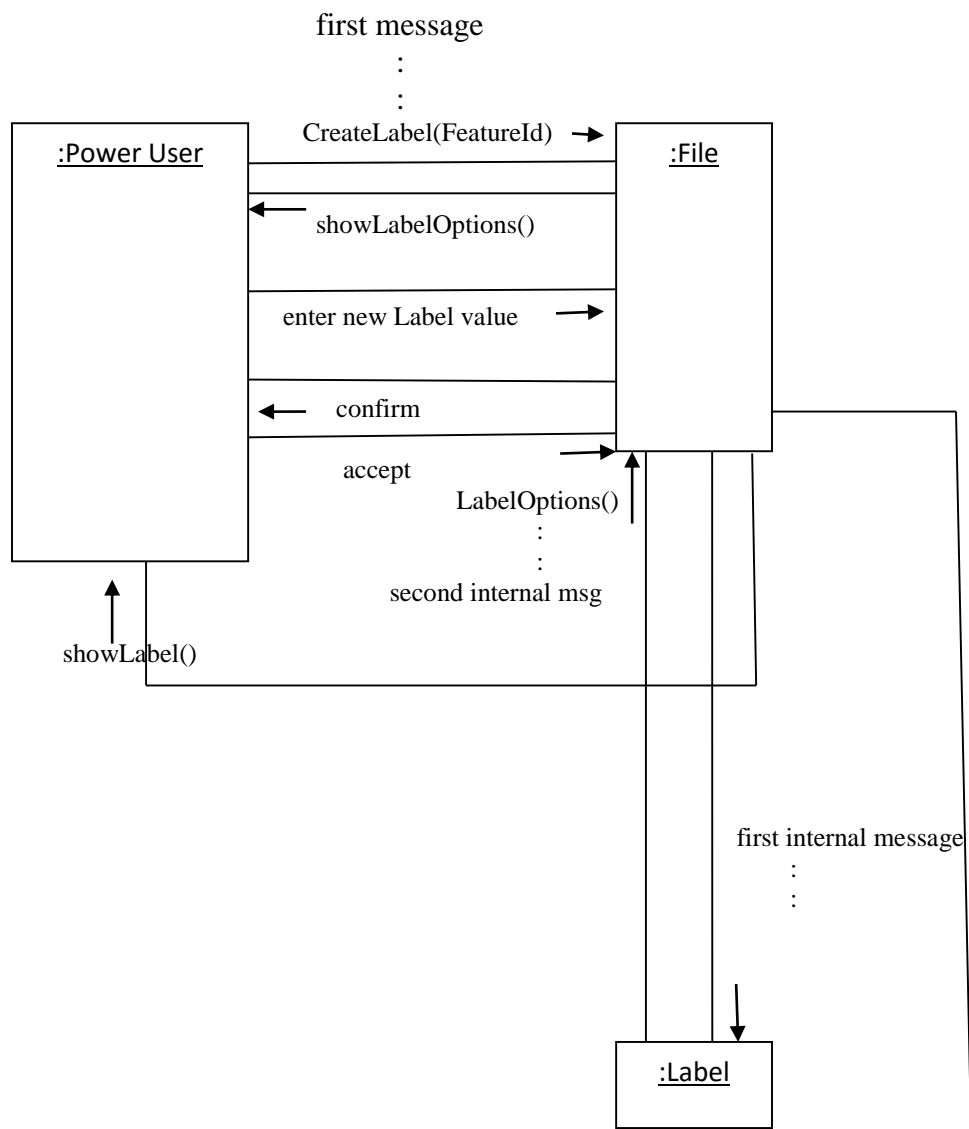
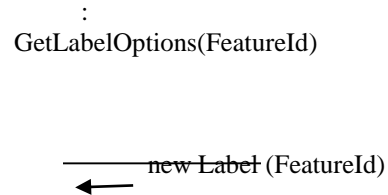


Figure: Collaboration Diagram for success scenario of Make Label use case





### 3.11.3 Functional Requirements

- REQ-1: User must be logged in as a Power user, show message to user if this is not the case.
- REQ-2: The file diagram must be already created. Show message to user notifying about she needs to create diagram before creating a new component.
- REQ-3: A Label can only be created for a given feature and not by itself.
- REQ-4: Each Feature can only have one Label, notify user of this fact.

## 3.12 Make Responsibility

### 3.12.1 Description and Priority

This creates a new Responsibility inside diagram. This use case is of high priority as it creates one of the main elements of our system.

- Benefit: 9
- Penalty: 3
- Cost: 3
- Risk: 2

### 3.12.2 Stimulus/Response Sequences

- User requests to create a Responsibility.
- System presents creation context to pass attributes values.
- User enters attribute values.
- System creates Responsibility, loads attributes with default values, places Responsibility in diagram in position indicated by attributes and shows Responsibility to the screen.

### 3.12.3 Functional Requirements

- REQ-1: User must be logged in as a Power user, show message to user if this is not the case.
- REQ-2: The file diagram must be already created. Show message to user notifying about she needs to create diagram before creating a new component.

- REQ-3: A Responsibility can be drawn inside or outside a Component. A line must pass over a responsibility.

### 3.13 Make Point

#### 3.13.1 Description and Priority

This feature creates a new Point inside diagram. This use case is of high priority as it is used to create the beginning and end points of a Line.

- Benefit: 9
- Penalty: 1
- Cost: 1
- Risk: 2

#### 3.13.2 Stimulus/Response Sequences

- System receives request to create a Point with a Boolean parameter (request is sent by the system itself when is creating a new Line)
- System creates a Point setting its “startPoint” attribute as indicated by passed parameters.
- System attaches Point to given Line (either the end or the beginning), places Point in diagram in position indicated by attributes and shows Point to the screen.

#### 3.13.3 Functional Requirements

- REQ-1: User must be logged in as a Power user, show message to user if this is not the case.
- REQ-2: The file diagram must be already created. Show message to user notifying about she needs to create diagram before creating a new component.
- REQ-3: A point can only exist for a given Line as either a beginning or end Point.
- REQ-4: A point should not be placed by itself in diagram, notify user of this error.

### 3.14 File Operations

#### 3.14.1 Description and Priority

We include here several file operations that will be high level descriptions of the operations supported by the operating system where the application will be installed. These features are: New File, Delete File, Save File, Open File and Close. Even though these operations are important, they are not essential to the overall function of the UCM system.

- Benefit: 6
- Penalty: 6
- Cost: 6
- Risk: 4

### 3.14.2 Stimulus/Response Sequences

Basically these operations rely on the OS to handle them.

- User requests to create a new file or delete or save or open or close a file.
- System receives the operation request, looks up a table that maps the operation with an OS method and calls the OS method that will handle the operation.
- The OS performs the request method and returns a result.
- The system receives the result and notifies the user with a success or failure message.

### 3.14.3 Functional Requirements








- REQ-1: A Power user can perform all file operations. User must be logged in as an Intermediate user to save a file. All users can open, and close a file. Show message to user indicating the operations allowed for each role.

## 4. External Interface Requirements

### 4.1 User Interfaces

The main focus of group A is to create a simple, robust, and consistent UCM Drawing tool. The GUI interface will consist of standard UI patterns and grouped standard buttons at the top of the screen. The UCM Drawing tool will verify and validate the user actions. In case of an error the UCM Drawing tool will produce window dialogs with error messages and a link to a help document.

Main Standard Buttons:

1. Component Button  which will allow the creation of components.
2. Responsibility Button  which will allow the creation of responsibility points inside the components.
3. Path Button  which will allow the creation of a path to/between components.
4. OR-fork Button  which will allow the OR-fork of paths.
5. OR-join Button  which will allow the OR-join of paths.
6. AND-fork Button  which will allow the AND-fork of paths.
7. AND-join Button  which will allow the AND-join of paths.

## **4.2 Hardware Interfaces**

The UCM Drawing tool will be Java based developed and will be cross platform supported under the Java Virtual Machine (JVM).

250MB of free disk space is required for the application installation and the data storage.

## **4.3 Software Interfaces**

The UCM Drawing tool is developed under Java 6 and therefore will require a compatible Java Runtime Environment (JRE) to be installed on the running machine.

The UCM Drawing tool will feature an export to image file (.jpg) which will create an image file of the drawing. An image viewer is required to open the images of the drawings.

## **4.4 Communications Interfaces**

The UCM Drawing tool will feature an open browser action to the URL of the software developer webpage.

# **5. Other Non-Functional Requirements**

## **5.1 Performance Requirements**

The UCM tool is a real-time application, thus the computation needs to minimize response time. All features directly related to diagram such as the ones presented in the uses cases need to be performed instantly.

## **5.2 Safety Requirements**

As a productivity software product, no special safety measure is necessary.

## **5.3 Security Requirements**

Individual diagrams are tied to a specific account. The user must be logged in to access and edit files.

## **5.4 Software Quality Attributes**

Software development focus will primarily be on the correctness of the product and if time allows it, the ease of testing.



## **6. Other Requirements**

No other requirements are to be noted for this project.

## **7. Appendix A: Glossary**

COCOMO: Constructive Cost Model

GUI: Graphical User Interface

KLOC: 1,000 Lines of Code

OS: Operating System

SRS: Software Requirements Specification

UCM: Use Case Maps

URL: Uniform Resource Locator (web address)

XML: Extensible Markup Language

## **8. Appendix B: Issues Lists**

No issues to report as of now.