# Unified Modeling Language

## 1 Problem statement

The purpose of this project is to create software in the form of a Unified Modeling Language that can be used to visualize the design of systems. It will be coded in Java using the JavaFX libraries and the design will consist of standard UML features such as the building classes and relationships between them. The software will be created at Millersville University by Abdul Elijah, Alayna Woleslagle, Brody Troutman, and Edward Rosensteel for Dr. Hutchens's CSCI 420 course.

## 2 System Personnel

### 2.1 Description of Users

Dr. Hutchens of Millersville University testing the software.

### 2.2 Description of System Developers

Abdul Elijah, Alayna Woleslagle, Brody Troutman, and Edward Rosensteel, students of Millersville University in CSCI 420.

## 3 Operational Setting

### 3.1 Target Platforms

Windows, Linux, and Mac that are capable of running a .jar file.

### 3.2 Required Software Environment

Latest version of java installed.

### 3.3 Useful Optional Software Environment

Additional UML's for comparison.

## 4 Functional Requirements

### 4.1 Functional Description

To create and edit visualizations of the design of systems through the use of visual elements of boxes, text, and lines in a user interface.

### 4.1.1 Overview

General purpose of system is to give standard options for creating and maintaining a UML diagram. The gray area presented in the software gives a drawing area to place the visual elements of boxes, relationships, and text.

The buttons at the top of the software give editing options to the user. New will create another window of the software with a blank drawing area. Save will save the positions of the elements in the drawing area to a text file. Open can read these files to place these elements onto the drawing area. Exit will close the software without saving. Help brings up a window that gives some instructions on what buttons do.

The buttons on the left side give the options for selecting which of these visual elements to add to the drawing area. Clicking the class box button will draw an instance immediately in the drawing area, text boxes work the same way. After clicking a relationship button, the user can click and drag on the drawing area to create the chosen relationship. The clear all button will bring up a warning window, when yes is clicked, the drawing area is reset. The show grid button will toggle a grid on the drawing area to assist with lining up diagrams, it can be clicked again to remove the grid.

To edit objects in the drawing they can be hovered over with the mouse to bring up editing elements. For class boxes the border around the box can be clicked and dragged to move the box. The small red box at the top left of the border can be clicked to delete the class box. The small green box at the bottom right of the border can be clicked to resize the class box, there is no size limit for the class box. The class box has 3 text fields that can be edited to add the proper title, attributes, and methods. The top title box has a fixed vertical size. Text boxes work the same way, but have a size limit. For relationships, when the line it hovered over it will turn red and can be clicked and dragged to move the relationship. The head of the relationship will also indicate a circle that must be clicked to rotate the relationship. Dragging elements will snap the corner of that element to the mouse.

### 4.1.2 Feature List

Input Selection

- Class Box: Creates a box in the drawing area with 3 sections to add text.
    - Adding Text: Text can be added to class boxes by clicking inside one of the 3 sections.
    - Moving: Clicking and dragging the red border around them.
    - Resizing: Clicking and dragging the green box at the bottom right of the border.
    - Deleting: Clicking the red box at the top left of the border.
- Choosing Relationship Type: Buttons that choose the relationship type that will show up on the drawing area next time the mouse is click and dragged there.
    - Aggregation: Solid line with white diamond tip.
    - Composition: Solid line with black diamond tip.
    - Generalization: Solid line with white arrow tip.

    - Dependency: Dotted line with lined arrow tip.
- Add Text Box: Creates a box in the drawing area that is a single section.
    - Adding Text: Text can be added to class boxes by clicking inside of them.
- Clear All: Removes all elements from drawing area.
- Show/Remove Grid: Toggles grid on drawing area.

Drawing Area: The gray area for placing chosen inputs, they will appear here and be editable.

Editing Buttons: The horizontal bar at the top of the interface, contains editing and miscellaneous buttons.

- New: Creates a new instance of the software.
- Save: Saves elements in drawing area to a text file.
- Open: Can load a text file to place elements on the drawing area.
- Exit: Closes the instance of the software.
- Help: Creates a new window that gives info about the software.

## 4.2 User Interface

Editing options are presented as buttons at the top of the window, selection options are presented as buttons on the left side of the window, and the area to place these selections is gray. Other interfaces are encountered when windows are opened to save and open files. The show/remove grid button is the only button that toggles its text. The "Exit" and "Help" buttons open new interfaces with buttons to warn and assist the user.

### 4.2.1 Overview

Placement of UI elements aligns with similar editing software standards. The buttons at the top of the software give editing options to the user. The buttons on the left side give the options for selecting which of these visual elements to add to the drawing area. To edit objects in the drawing they can be hovered over with the mouse to bring up editing elements.

### 4.2.2 Menus

- Editing buttons
- Input selection buttons

### 4.2.3 Windows

- The majority of the software is in one window with the exception of the "Help" button that creates an info window.
- New windows can be created with the "New" button
- File selection windows are opened for the "Save" and "Open" buttons

### 4.2.4 Inspectors

## 4.3 Use Cases

### 4.3.1 Use Case 1: Add class box to UML display area

Precondition: The UML application should be open.

- User selects the “Class Box” button from the side menu.
- System draws class box in the default location.

Postcondition: A new class box has been placed in the display area.

### 4.3.2 Use Case 2: Create an aggregation relationship between two class boxes

Precondition: The UML application should be open and two class boxes should already be displayed on screen(See Use Case 1).

- User selects line type from buttons. System should display all line types available.
- From the given options the user selects Aggregation. System selects the desired line type.
- User clicks and drags the mouse pointer from one class box to another. System should display a line from the location where mouse was clicked to the location where the click is released.

Postcondition: A line should be drawn between the two class boxes representing a aggregation relationship.

### 4.3.3 Use Case 3: Create a composition relationship between two class boxes

Precondition: The UML application should be open and two class boxes should already be displayed on screen(See Use Case 1).

- User selects line type from buttons. System should display all line types available.
- From the given options the user selects Composition. System selects the desired line type.
- User clicks and drags the mouse pointer from one class box to another. System should display a line from the location where mouse was clicked to the location where the click is released.

Postcondition: A line should be drawn between the two class boxes representing a composition relationship.

### 4.3.4 Use Case 4: Create a generalization relationship between two class boxes

Precondition: The UML application should be open and two class boxes should already be displayed on screen(See Use Case 1).

- User selects line type from buttons. System should display all line types available.
- From the given options the user selects Generalization. System selects the desired line type.
- User clicks and drags the mouse pointer from one class box to another. System should display a line from the location where mouse was clicked to the location where the click is released.

Postcondition: A line should be drawn between the two class boxes representing a generalization relationship.

### 4.3.5 Use Case 5: Create a dependency relationship between two class boxes

Precondition: The UML application should be open and two class boxes should already be displayed on screen(See Use Case 1).

- User selects line type from buttons. System should display all line types available.
- From the given options the user selects Dependency. System selects the desired line type.
- User clicks and drags the mouse pointer from one class box to another. System should display a line from the location where mouse was clicked to the location where the click is released.

Postcondition: A line should be drawn between the two class boxes representing a dependency relationship.

#### 4.3.6 Use Case 6: Add class box to UML display area

Precondition: The UML application should be open.

- User selects the "Add Text" button from the side menu.
- System draws text box in the default location.

Postcondition: A new text box has been placed in the display area.

#### 4.3.7 Use Case 7: Clear all objects from drawing area

Precondition: The UML application should be open and any number of class boxes, text boxes, or relationships are placed in the drawing area. (See Use Case 1)

- User selects the "Clear All" button from the side menu.
- System displays a warning message.
- User selects the "Yes" button.
- System deletes all objects in the drawing area.

Postcondition: The display area is clear of all objects.

#### 4.3.8 Use Case 8: Enable grid for the drawing area

Precondition: The UML application should be open.

- User selects the "Show Grid" button from the side menu.
- System changes button display text to "Remove Grid"
- System displays a grid in the drawing area.

Postcondition: The display area displays a grid.

#### 4.3.9 Use Case 9: Disable grid for the drawing area

Precondition: The UML application should be open.

- User selects the "Remove Grid" button from the side menu.
- System changes button display text to "Show Grid"
- System removes the grid in the drawing area.

Postcondition: The display area returns to default.

#### 4.3.10 Use Case 10: Close UML application

Precondition: The UML application should be open.

- User selects the "exit" button.
- System should display a pop up window warning user to save before exiting.
- User selects "exit and close application" button. System should close entire window.

Postcondition: UML application should be closed

#### 4.3.11 Use Case 11: Saving a UML diagram

Precondition: The UML application should be open and any number of class boxes, text boxes, or relationships are placed in the drawing area. (See Use Case 1)

- User selects the "Save" button.
- System should display a pop up window prompting the user to select a directory and file name.
- User selects "Save".
- System should create a text file with the coordinates of the objects in the diagram.

Postcondition: A save file is created for the diagram

#### 4.3.12 Use Case 12: Saving a UML diagram

Precondition: The UML application should be open and the drawing area is empty.

- User selects the "Open" button.
- System should display a pop up window prompting the user to select a directory and file name.
- User selects "Open".
- System should create a diagram with the coordinates of the objects from the text file.

Postcondition: A file is loaded, and its objects and coordinates are displayed on the drawing area.

# 5 Non-functional Requirements

## 5.1 Reliability

Able to place and combine all different input selection variations without glitches, slow down, crashes, etc.

## 5.2 Performance

Responsive startup, output visualization, editing and selecting operations, and shutdown.

## 5.3 Usability

Clear drawing area and matching expected operations with displays of selection and editing area.

## 5.4 Portability

Runnable on any Linux, Windows, or Mac machine that has the latest version of Java installed.

# 6 Future Enhancements

- Further organization and separation of code into classes
- Controllers for objects
    - Selection interface UI for controllers

        - Shows variables of selected object
        - Can edit variables
- Horizontal and vertical snapping
- Drawing area scrollable