SYSTEMS AND SOFTWARE REQUIREMENTS SPECIFICATION (SSRS) FOR

Phunctional UML Editor
(pUML)

Version 1.0
May 8, 2012

Prepared for:
Dr. Clint Jeffery

Prepared by:
Josh Armstrong
Zach Curtis
Brian Bowles
Logan Evans
Jeremy Klas
Nathan Krussel
Maxine Major
Morgan Weir
David Wells
University of Idaho
Moscow, ID 83844-1010

pUML SSRS

RECORD OF CHANGES

Change Number	Date	Location of change (e.g., page or figure #)	A M D	Brief description of change	Initials
1	12/10/2011	Section 3	A	Text material for section 3 of the SSRS document.	LE
2	01/17/2012	SSRS Document	A	Added updated SSRS/SSDD pdf and TeX files	MM
3	02/01/2012	SSRS Document	M	Updated SSRS	MM
4	02/15/2012	Section 3	M	Use cases expanded and descriptions added	MM
5	02/23/2012	Section 3	M	Use case descriptions completed	MM
6	04/04/2012	Section 3	M	Revised use case descriptions	MW
7	04/09/2012	Section 2	A	Requirements update	MM
8	04/23/2012	Section 2.1	A	Incorporated Test Plan breakdown of requirements	MM
9	04/30/2012	Section 2.1	M	Update reqs traceability section	MM
10	04/30/2012	Section 4	M	Merge restore and open feature descriptions	MM
11	04/30/2012	Section 3.2.1	M	Update all use cases and descriptions	MM
12	05/07/2012	Section 3.2	M	Modify use cases	MM
13	05/08/2012	Section 3.2	M	Incorporate previous section 3.2.2 into 3.2.1 sections	MM

*__A__ - ADDED __M__ - MODIFIED __D__ - DELETED

pUML SSRS
TABLE OF CONTENTS

1 Introduction

1.1 IDENTIFICATION

The software system being considered for development is referred to as Phunctional UML Editor (pUML). The customer providing specifications for the system is Dr. Clint Jeffery at the University of Idaho. The ultimate customer, or end-user, of the system will be University of Idaho Computer Science students and/or faculty. This is a new project effort, so the version under development is version 0.9.
Version 1.0 will be completed upon pUML release date.

1.2 PURPOSE

The purpose of the system under development is to create and store UML diagrams.
The pUML software will be used by Computer Science students at the University of Idaho, and is intended to be read and understood by the same.

1.3 SCOPE

The pUML software was conceptualized as a Software Engineering class project, and following its launch in September 2011, spanned nine months and a total of ten software engineering students. The pUML project will be considered finalized as of May 10, 2012, and will not have aquirers or support agencies as of the date of release.

Upon completion, the pUML software will be available only for distribution to the University of Idaho Computer Science department. The pUML software will no longer be supported by the initial software development team, and will be supported only at the direction of University of Idaho Computer Science faculty.

1.4 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

Term or Acronym	Definition
Alpha test	Limited release(s) to selected, outside testers
Beta test	Limited release(s) to cooperating customers wanting early access to developing systems
Final test	aka, Acceptance test, release of full functionality to customer for approval
SSDD	System and Software Design Document
SSRS	System and Software Requirements Specification

1.5 REFERENCES

The pUML requirements will be addressed in the following documents:

- Test Plan (TP), Version 1.0
- System and Software Design Description (SSDD) Version 1.0

1.6 OVERVIEW AND RESTRICTIONS

This document is for limited release only to University of Idaho Computer Science personnel assigned to the pUML project.

Section 2 of this document describes the system under development from a holistic point of view. Functions, characteristics, constraints, assumptions, dependencies, and overall requirements are defined from the system-level perspective.

Section 3 of this document describes the specific requirements of the system being developed. Interfaces, features, and specific requirements are enumerated and described to a degree sufficient for a knowledgeable designer or coder to begin crafting an architectural solution to the proposed system.

Section 4 provides the requirements traceability information for the project. Each feature of the system is indexed by the SSRS requirement number and linked to its SDD and test references.

2 OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

This product is independent of any other product, and as such, is self-contained.

2.2 PRODUCT FUNCTIONS

This product's primary function is to allow the user to create UML diagrams, edit existing diagrams, and store and retrieve them.
Features to be tested include all objects, connectors, and associated functionality, Open/Save functionality, and all diagram types load properly. The specific requirements for the UML diagram editor are itemized insubsequent sections.

2.2.1 INSTALL / LAUNCH / EXIT

Basic functionality that must be met prior to use of this software involves the successful installation and launch of the pUML software:

- pUML software includes an installer, which at basic functionality will:
  - Consist of a single downloadable installation file.
  - Install on Windows and Linux platforms, resulting in a single application icon.
  - Uninstall from both Windows and Linux platforms.
- Once installed, the pUML software may be launched by double-clicking the pUML icon.
- pUML may be exited by clicking an "X" in the upper corner of the software, or by selecting "File," and then "Exit."

2.2.2 MENU OPTIONS / MAIN WINDOW

- Options window appears at pUML startup.
- When the options window is closed with no selections made, the pUML main window remains.
- When "New" is selected, options window appears.
- When "Open" is selected, an explorer window appears.
- When "Save" is selected: refer to subsection 2.1.6.
- When "Save As.." is selected: refer to subsection 2.1.6.
- When Exit is clicked:
  - If all currently open diagrams are saved, pUML exits.
  - If any open diagram is not saved, refer to subsection 2.1.6.

2.2.3 OBJECTS

All objects should perform or have the following tasks performed on them in a consistent manner in the following cases:

- Objects can be created.
- Objects can be deleted.
  - Any associated connectors must also be deleted.
  - All associated text and descriptions must also be deleted.
- Objects can be selected.
- Objects can be de-selected.
- Objects can be moved after they have been placed
  - Objects moved retain the new location after the associated diagram is saved, closed, and restored.
  - All text and descriptions associated with the object move with the object to the same location in relation to the object.
- Objects can accept an indefinite number of connection lines.
- The object's properties may be modified an indefinite number of times.
- The object will automatically resize to fit the contents.

2.2.4 CONNECTORS

All connectors should perform or have the following tasks performed on them in a consistent manner in the following cases:

- Connectors can be created.
  - Only one connector may exist between any two objects.
  - Only one self connector may exist per object.
- Connectors can be deleted.
- Connectors can be selected.
- Connectors can be de-selected.
- Connectors cannnot be moved from one object and placed onto another.
- If an object a connector is attached to moves, the connector's endpoint attached to that object will move with that object.
  - Connectors moved retain the new configuration with the same integrity as the previous configuration.
  - All text and descriptions associated with the connector move with the connector to the same location in relation to the connector as before.
- The connector's properties may be modified.
  - Line description may be modified an indefinite number of times.

2.2.5 OPEN

Once the pUML software has been launched, files may be opened if the following criteria are met:

- The file type is of pUML type.
- The file name is legitimate.
- The file is not already open in pUML.

Note: The cases listed above are ensured by QT, and will not be tested.

If the user opens a pUML diagram file, restore functionality will occur as follows:

- If the pUML software is not already loaded, it will load.
- A new tab will open, and the previously saved diagram will appear in the tab.
- The toolbar is updated with that diagram's objects and connectors.
- All objects and connectors in the file are in the exact same state and configuration as when the file had been previously saved.
- The file may be modified (edited).
  - Objects may be added or removed.
  - Connectors may be added or removed.
- The file may be saved.
- The file may be closed.

2.2.6 SAVE/SAVE AS

The Save and Save As functions may be invoked if the following criteria are met:

- A diagram is open in pUML.

Save/Save As functionality:

- A diagram may be saved regardless of whether it is new or previously saved.
- A diagram may be saved whether or not it has been modified. (e.g., a blank diagram may be saved)
- If a diagram has been previously saved, pUML will automatically write over the saved file's contents. The modified diagram will be saved at the original location of the diagram under the original name.
- If a diagram has not been previously saved, Save As will automatically occur.
  - The Save-As dialog will load.
  - The user must select an appropriate file name and storage location.
  - The diagram will be stored at the location and under the name the user has entered.

2.2.7 DIAGRAM TYPES

- pUML diagrams may be of only one diagram type. (There are no legitimate UML diagrams that combine elements of more than one diagram type)
- Diagram-specific objects and connectors will be the only ones displayed in the toolbar for a specific diagram type.
- The type of the diagram will be saved with a diagram file, so that a diagram may not be loaded into a tab with a different diagram type.

2.2.8 TAB FUNCTIONALITY

- Except for the initial loading of pUML (initial options window view), at least one tab must be open at any time.
- If all tabs are closed, pUML exits.
- If the diagram loaded into a tab is saved, clicking the "X" will close the tab.
- If the diagram loaded into a tab is not saved, and the "X" is clicked: refer to subsection 2.1.6.
- pUML will permit an indefinite number of tabs to be opened at any time, but this may consume a disproportionate amount of resources.

2.3 USER CHARACTERISTICS

The intended user for the pUML software are software engineers and computer science students with a prior knowledge of UML. This user is already familiar with computers and generally has some experience in programming languages.

2.4 CONSTRAINTS

Since the pUML project was developed as a class assignment, further development of this project will halt if the University of Idaho faculty overseeing this project decide that this project should to no longer continue.

2.5 ASSUMPTIONS AND DEPENDENCIES

The requirements for the pUML software were decided upon by University of Idaho Computer Science Department faculty. Any further direction this project may take will depend on their decisions. As of this revision, it is currently undecided if this project will be maintained after the corresponding course has concluded.

2.6 SYSTEM LEVEL (NON-FUNCTIONAL) REQUIREMENTS

2.6.1 Site dependencies

The pUML software has no dependencies on any external resources, such as internet access, etc.. Any modern operating system (2008+) should be sufficient to support the pUML software, and since this software is cross-platform, there should be no complications.

2.6.2 Safety, security and privacy requirements

There are no safety, security or privacy requirements at this time.

2.6.3 Performance requirements

This software is to be supported on one terminal per install, and since there are no current limitations applying to product distribution, it may be installed on a theoretically infinite number of terminals. This software is not designed to be remotely accessed, and as a result, one user per session is recommended as well. The software has not been tested to determine efficient transaction times as of the date of this SSRS publication.

2.6.4 System and software quality

The fully developed software should be available for use and reliably handle all requests 98 percent of the time. The software can currently handle saving, diagram selection, allow for editing of multiple diagrams at a time, diagram component deletion, diagram component labeling, and loading of saved diagrams.

2.6.5 Packaging and delivery requirements

The executable system and all associated documentation (i.e., SSRS, SSDD, code listing, test plan (data and results), and user manual) will be delivered to the customer on CD's and/or via email, as specified by the customer at time of delivery. Although the user will be kept informed of progress via the Mercurial repository throughout the system development process, the final, edited version of the above documents will accompany the final, accepted version of the executable system.

2.6.6 Personnel-related requirements

The system under development has no special personnel-related characteristics.

2.6.7 Training-related requirements

A user manual will be included in this project to aid users who have difficulty intuiting the proper action to produce the corresponding outcome.

2.6.8 Logistics-related requirements

The pUML software is intended for use on University of Idaho Computer Science department computers as well as computer science students' personal computers including, at a minimum, operating systems Windows 7, Mac OSX, and Linux. Any minimum hardware requirements lie outside the scope of the resources available, and there will be no software application dependencies at the time of release.

2.6.9 Precedence and criticality of requirements

System and software quality are the primary focus of our software development. All additional features and functionality are secondary.

3 SPECIFIC REQUIREMENTS

3.1 EXTERNAL INTERFACE REQUIREMENTS

3.1.1 Hardware Interfaces

Any computing system capable of running Linux or Windows 7.

3.1.2 Software Interfaces

Either a Linux or Windows 7 operating system to guarantee functionality.

3.1.3 User Interfaces

The only user interfaces required to operate the pUML software are standard computing interfaces

- Monitor
- Keyboard
- Mouse

3.1.4 Other Communication Interfaces

No other interfaces are required.

3.2 SYSTEM FEATURES

3.2.1 Use Cases and Descriptions

The features detailed in the following use cases are all basic vital functions for a UML Diagram editor. At the most basic level, in order to create UML diagrams, several processes are required. These processes include the creation of distinct UML diagrams consisting of labeled objects and connectors, saving of these diagrams, and the ability to load these saved diagrams for modification.

3.2.1.1 Launch These options will be available to the user upon the launch of pUML.

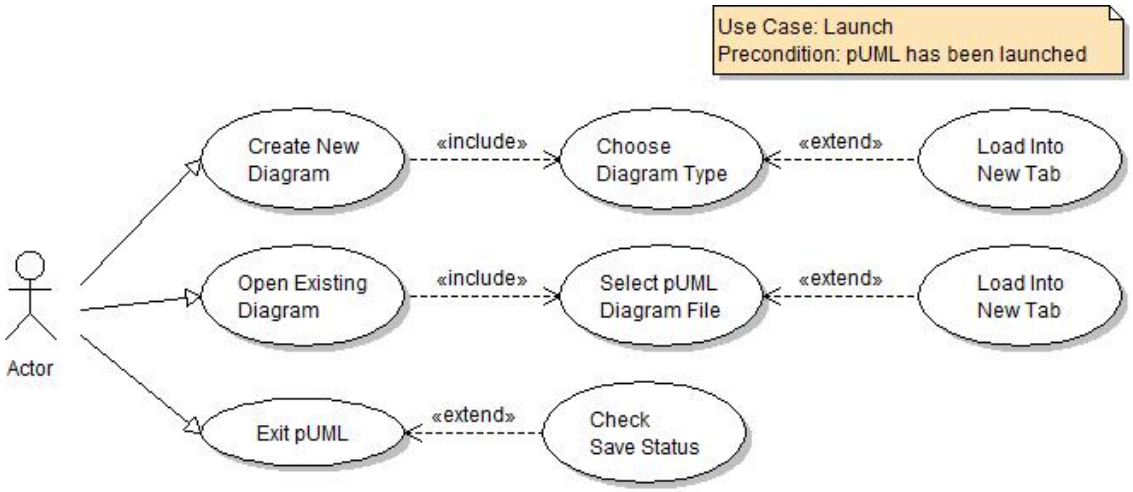

Figure 1: Launch Use Case

3.2.1.1.1 Create New Diagram This creates a new canvas in which the user may design a UML diagram. Since there are four different diagram types that may be created with pUML, and many design projects necessitate several diagrams, there is currently no predetermined limit on the number of new diagrams that may be created in pUML. (This is determined by system resources only, and may differ from system to system.)

Use Case Name	**Create New Diagram**
Participating Actor	User
Flow of Events	1. The user selects "New" from menu. 2. Program requests user select a diagram type. 3. The program responds by creating a new tab with a blank drawing canvas. 4. Program loads and displays only objects for the selected diagram type.
Entry Condition	User selects "New" from commands
Exit Condition	Program successfully opens new tab with diagram specific tools in toolbar.
Quality Requirements	Toolbar loads only diagram specific objects and connectors.

3.2.1.1.2 Open Existing Diagram This opens a previously saved UML diagram. Since each diagram is of a specific diagram type, when a pUML diagram is loaded, only the tools available for that diagram type will be available to the user upon loading. When a diagram has been opened, the user then may modify this diagram further. There is no limit to the number of times a diagram may be modified. There is currently no limit to the number of diagrams that may be opened into the pUML software. (This is determined by system resources only, and may differ from system to system.)

Use Case Name	**Open Existing Diagram**
Participating Actor	User
Flow of Events	1. User selects "Open" from menu. 2. Program opens an explorer window. 3. User selects the file they wish to open. 4. Program loads selected file into a new tab.
Entry Condition	User selects "Open" from menu.
Exit Condition	File is successfully opened into a new tab.
Quality Requirements	User selected file must be of pUML diagram type. Only diagram-appropriate tools loaded into toolbar.

3.2.1.1.3 Exit pUML The user may exit pUML by either selecting "File" and then "Exit," or may click the "X" in the upper corner of the pUML main window. If any currently loaded diagrams have changes which have not been saved, pUML will request confirmation to either save or discard changes prior to exiting.

Use Case Name	**Exit pUML**
Participating Actor	User
Flow of Events	1. User clicks X. 2. If a file is open, program checks to see if it is saved. If no files are open, simply exits. 3. Program Exits.
Entry Condition	User initiates File menu option "Close Program" or clicks X in the corner of the window.
Exit Condition	File is successfully saved and the program exits.
Quality Requirements	File must be saved prior to exiting, or changes must be discarded prior to exiting. If the current tab closed was the only tab open in pUML, pUML exits.

3.2.1.2 New/Open Diagram These options will be available to the user after creating a new UML diagram, or upon opening an existing UML diagram. This set of use cases covers all edits possible for pUML diagrams.

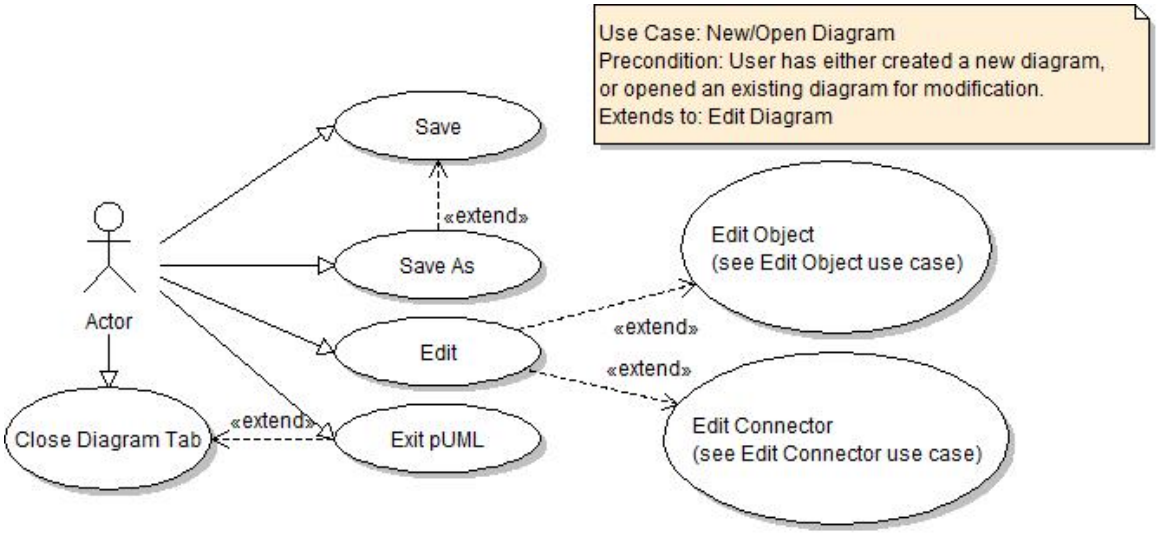

Figure 2: New/Open Diagram Use Case

3.2.1.2.1 Save This saves the current state of the diagram into an XML file that can later be used to recreate the diagram state. If the current diagram does not have a name, the "Save" option will be an alias for the "Save As" option.

Use Case Name	**Save**
Participating Actor	User
Flow of Events	1. The user opens the File menu and selects "Save." 2. Program saves file under current file name (a) If file has not been previously saved, program initiates Save As use case. (b) If file has been previously saved, the program saves the file at its existing location under its existing name.
Entry Condition	User selects "Save File" from main menu or clicks Save button.
Exit Condition	File is successfully saved
Quality Requirements	File names must be valid May not create two pUML diagram files with same name in same location.

3.2.1.2.2 Save As This prompts pUML to load a browser window. The user must select a location and a file name in which to store the pUML diagram. After clicking "Ok," the Save function completes this process. This state will be defaulted to if the user clicks "Save" on a previously unsaved diagram.

Use Case Name	**Save As**
Participating Actor	User
Flow of Events	1. User opens File menu and selects "Save As..." or 1. User selects "Save" from File menu, but the program has not been previously saved. 2. Program displays a dialog box requesting the user to enter a diagram name and save location. 3. User enters a diagram name and clicks "Ok." 4. Program saves diagram at user specified location.
Entry Condition	User clicks "Save As" or clicks "Save" with a previously unsaved diagram.
Exit Condition	The diagram has been saved.
Quality Requirements	Inherited from Save use case.

3.2.1.2.3 Close Diagram Tab If the user clicks the "x" on the tab, and if the current diagram state is saved, the diagram will close. Otherwise, the "Save As" function will initiate prior to closing. This feature is useful to de-clutter the pUML workspace. In addition, if the closed tab was the only tab open, then the pUML software will exit.

Use Case Name	**Close Diagram Tab**
Participating Actor	User
Flow of Events	1. User clicks the 'X' on the diagram tab. 2. If the current version of the diagram is saved, the program closes the diagram tab. or 1. User clicks the 'X' on the diagram tab. 2. If the program is not saved, the program issues a warning.
Entry Condition	User clicks the "X" on the diagram tab.
Exit Condition	Diagram tab is closed.
Quality Requirements	Diagram must be saved for the tab to close, or the user must either choose to save the diagram or discard changes.

3.2.1.3 Use Objects These options will be available to the user with a diagram open for modification, who is choosing to take action on an object.

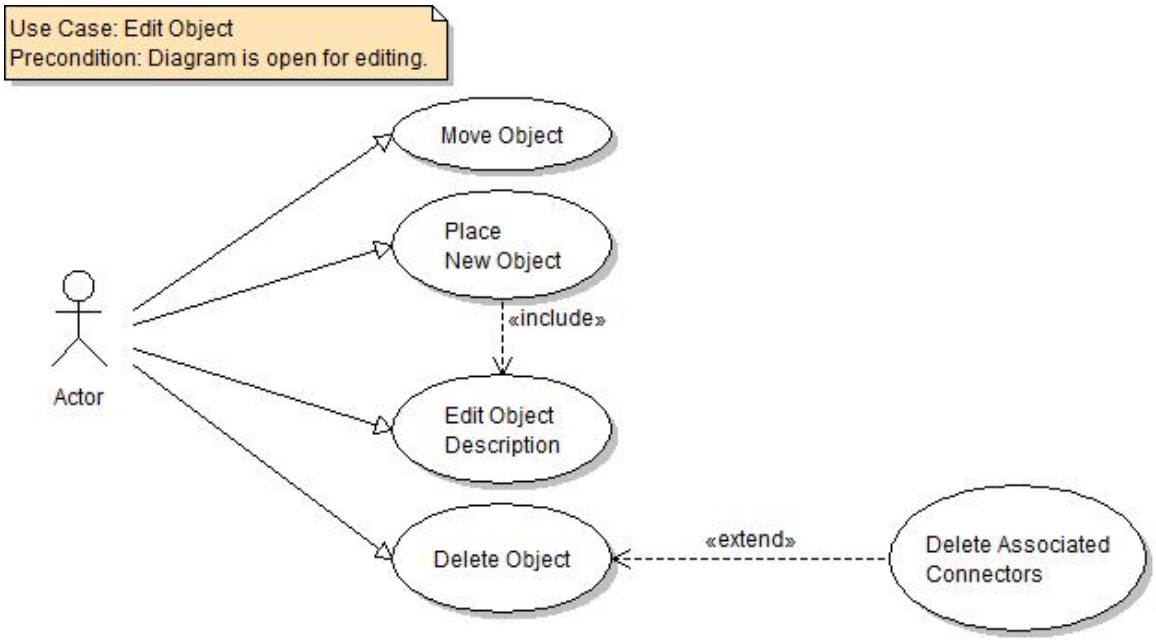

Figure 3: Edit an Object Use Case

3.2.1.3.1 Place New Object

Use Case Name	**Place New Object**
Participating Actor	User
Flow of Events	1. User selects an object from toolbar. 2. User clicks the canvas. 3. Program places an object at that location on the canvas. 4. User may be prompted to enter a description for the object.
Entry Condition	pUML diagram is open for editing. Toolbar is loaded with valid objects for diagram type.
Exit Condition	Object has been successfully placed on drawing canvas.
Quality Requirements	Object must be of diagram type.

3.2.1.3.2 Edit Object

Use Case Name	**Edit Object Description**
Participating Actor	User
Flow of Events	1. User selects an object by clicking on it. 2. User right-clicks on the object. 3. User selects "Properties" from the drop-down menu. 4. User types the desired description in the text box presented. 5. User clicks "OK." 6. Program displays new description.
Entry Condition	User right-clicks on an object and selects "Properties" from the dropdown menu.
Exit Condition	Object now displays new description.
Quality Requirements	N/A

3.2.1.3.3 Move Object

Use Case Name	**Move Object**
Participating Actor	User
Flow of Events	1. User clicks on object, selecting it. 2. User holds down the left mouse button, and moves the mouse to the new location of the object. 3. User releases the left mouse button. 4. Program draws the object at the new location, with all attached connectors.
Entry Condition	User selects an object with the left mouse button held down.
Exit Condition	Program has drawn the object at the new location.
Quality Requirements	Connectors move with object.

3.2.1.3.4 Delete Object

Use Case Name	**Delete Object**
Participating Actor	User
Flow of Events	1. User right clicks object and selects "Delete" from the dropdown menu. 2. Program deletes the object and all attached connectors.
Entry Condition	User right clicks object and selects "Delete" from the dropdown menu.
Exit Condition	Program deletes object and all attached connectors.
Quality Requirements	Attached connectors must be deleted with deleted object.

3.2.1.4 Use Connectors These options will be available to the user with a diagram open for modification, who is choosing to take action regarding a connector.

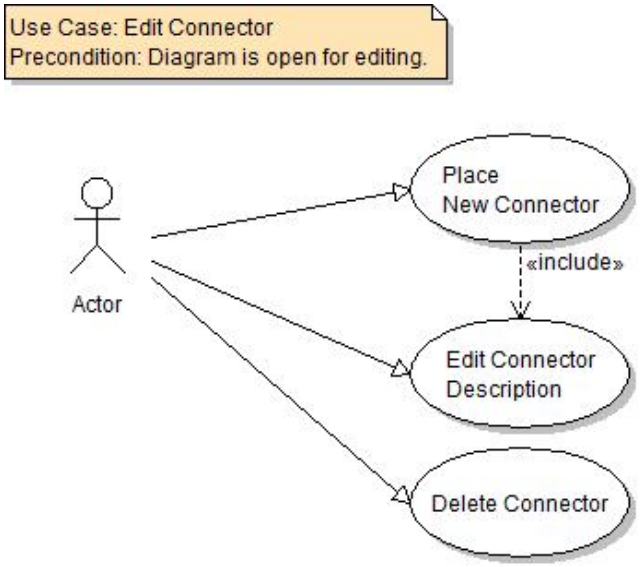

Figure 4: Edit a Connector Use Case

3.2.1.4.1 Place New Connector

Use Case Name	**Place New Connector**
Participating Actor	User
Flow of Events	1. User clicks on desired connector in toolbar. 2. User clicks on the first (origination) object, holding down the left mouse button. 3. User drags the mouse to a second (destination) object. 4. User releases the mouse button. 5. Program draws the connection line between the first and second object.
Entry Condition	User has selected a connector from the toolbar.
Exit Condition	The connector is successfully drawn between two objects.
Quality Requirements	Only one connector may be placed between any two objects.

3.2.1.4.2 Edit Connector

Use Case Name	**Edit Connector Properties**
Participating Actor	User
Flow of Events	1. User right-clicks on the connector and selects "Properties" from the drop-down menu. 2. User edits description in the text box presented. 3. User clicks "OK." 4. Program displays new line description.
Entry Condition	User right-clicks on the connector and selects "Properties" from the drop-down menu.
Exit Condition	Program has updated the connector with the new properties.
Quality Requirements	Some connectors may not have the option to edit/modify the description.

3.2.1.4.3 Delete Connector

Use Case Name	**Delete Connector**
Participating Actor	User
Flow of Events	1. User right clicks on a connector and selects "Delete" from the drop-down menu. 2. Program deletes the connector.
Entry Condition	User right clicks on a connector and selects "Delete" from the drop-down menu.
Exit Condition	Connector has been deleted.
Quality Requirements	Only the connector is to be deleted. Attached objects are to remain on the canvas.

4 REQUIREMENTS TRACEABILITY

Feature Name	Req No.	Requirement Description	Priority	SSDD	Test Case(s)	Alpha Release	Beta Release	Final Test
Install	2.2.1	pUML software installs successfully	M	N/A	testinstall	PASS	PASS	PASS
Launch	2.2.1	pUML software launches successfully	M	N/A	testlaunch	PASS	PASS	PASS
Exit	2.2.1	pUML Exit options	M	4.3.3	testsave * testsaveas *	FAIL FAIL	PASS PASS	PASSPASS
Main Window	2.2.2	All main window features	M	*N/A	testmainwindow	PASS	PASS	
Objects	2.2.3	Object properties and functionality	M	4.3.7 4.3.8 4.3.9 4.3.10	testobjects	PASS	PASS	
Connectors	2.2.4	Connector properties and functionality	M	4.3.11 4.3.12 4.3.13	testconnectors testselfconnector	FAIL PASS	PASS PASS	PASS
Open	2.2.5	Load a pUML file into pUML	M	4.3.2	testsave * testsaveas *	FAIL FAIL	PASS PASS	
Save/Save As	2.2.6	Save a diagram	M	4.3.4 4.3.5	testsave testsaveas	FAIL PASS	PASS PASS	
Diagrams	2.2.7	Diagram type integrity	M	4.3.1	testsave * testsaveas *	FAIL FAIL	PASS PASS	
Tabs	2.2.8	Tab functionality	M	4.3.6	testsave * testsaveas *	FAIL FAIL	PASS PASS	

Priorities are: **M**andatory, **L**ow, **H**igh
SSDD link is version and page number or function name.
Test cases and results are file names and **P**ass/**F**ail or % passing.
* asterisk indicates indirectly tested through another feature test case.