



**SYSTEMS AND SOFTWARE REQUIREMENTS SPECIFICATION (SSRS) FOR**  
**Phunctional UML Editor**  
**(pUML)**

**Version 0.9**  
**April 30, 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**

<b>Change Number</b>	<b>Date</b>	<b>Location of change (e.g., page or figure #)</b>	<b>A M D</b>	<b>Brief description of change</b>	<b>Initials</b>
1	12/10/2011	Section 3	A	Text material for section 3 of the SSRS document.	LE
2	01/17/2012		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 Section 4	M	Update reqs traceability section Merge restore and open feature descriptions	MM

**\*A - ADDED M - MODIFIED D - DELETED**

**PUML SSRS  
TABLE OF CONTENTS**

**Section   Page**

# 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 7, 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 2.0
- System and Software Design Description (SSDD) Version 0.0

## **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.

#### **2.1.1 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 below.

#### **2.1.2 MENU OPTIONS MENU MAINWINDOW**

- 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.

#### **2.1.3 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.

#### **2.1.4 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 cannot 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.

#### **2.1.5 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.

#### **2.1.6 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.

#### **2.1.7 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.



### **2.1.8 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.

## **2.2 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.

## **2.3 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 no longer continue.

## **2.4 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.

## **2.5 SYSTEM LEVEL (NON-FUNCTIONAL) REQUIREMENTS**

### **2.5.1 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.

### **2.5.2 Safety, security and privacy requirements**

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

### **2.5.3 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.

#### **2.5.4 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.

#### **2.5.5 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.

#### **2.5.6 Personnel-related requirements**

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

#### **2.5.7 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.

#### **2.5.8 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.

#### **2.5.9 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 following use cases represent different stages at which options will be available to the user.

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

USE CASE DESCRIPTION GOES HERE

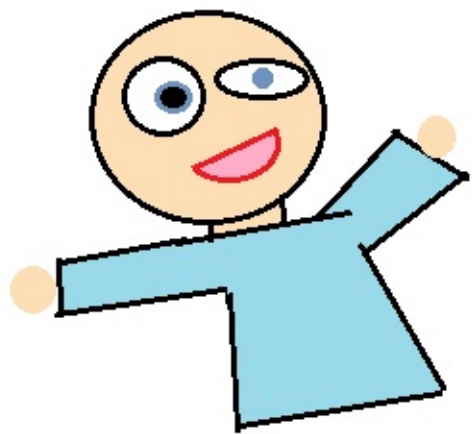


Figure 1: Launch Use Case

<i>Use Case Name</i>	Exit Program
<i>Participating Actor</i>	User
<i>Flow of Events</i>	<div>1. User clicks X.</div> <div>2. If a file is open, program checks to see if it is saved. If no files are open, simply exits.</div> <div>3. Program Exits.</div>
<i>Entry Condition</i>	User initiates File menu option “Close Program” or clicks X in the corner of the window.
<i>Exit Condition</i>	File is successfully saved and the program exits.
<i>Quality Requirements</i>	File must be saved prior to exiting, or changes must be discarded prior to exiting.

**3.2.1.2 New Project** The user has selected to start a new UML project.

<i>Use Case Name</i>	<b>Create New Diagram</b>
<i>Participating Actor</i>	User
<i>Flow of Events</i>	<ol style="list-style-type: none"><li>1. The user selects new file from menu.</li><li>2. Program requests user select a diagram type (includes ChooseDiagramType use case).</li><li>3. The program responds by creating a new file with a blank drawing canvas.</li></ol>
<i>Entry Condition</i>	User selects New File from commands
<i>Exit Condition</i>	Program successfully opens new file
<i>Quality Requirements</i>	TBD

<i>Use Case Name</i>	<b>Choose Diagram Type</b>
<i>Participating Actor</i>	User
<i>Flow of Events</i>	<ol style="list-style-type: none"><li>1. User selects a diagram type.</li><li>2. Program loads and displays only objects for the selected diagram type.</li></ol>
<i>Entry Condition</i>	Included from New Diagram use case
<i>Exit Condition</i>	Included from New Diagram use case
<i>Quality Requirements</i>	Toolbar loads only diagram specific objects and connectors.

**3.2.1.3 Open Project** These options will be available to the user after choosing to open an existing UML project.

<i>Use Case Name</i>	<b>Open Existing Diagram</b>
<i>Participating Actor</i>	User
<i>Flow of Events</i>	<ol style="list-style-type: none"><li>1. User selects Open File from menu.</li><li>2. Program opens an explorer window.</li><li>3. User selects a file.</li><li>4. Program loads selected file into program</li></ol>
<i>Entry Condition</i>	User selects Open File from menu.
<i>Exit Condition</i>	File is successfully opened.
<i>Quality Requirements</i>	User selected file must be able to be opened in pUML.

**3.2.1.4 New Diagram** These options will be available to the user upon choosing to create a new UML diagram.

<b><i>Use Case Name</i></b>	<b>Save File</b>
<b><i>Participating Actor</i></b>	User
<b><i>Flow of Events</i></b>	<ol style="list-style-type: none"> <li>1. The user opens the File menu and selects “Save.”</li> <li>2. Program saves file under current file name <ol style="list-style-type: none"> <li>(a) If file has not been previously saved, program initiates Save As use case.</li> <li>(b) If file has been previously saved, the program saves the file at its existing location under its existing name.</li> </ol> </li> </ol>
<b><i>Entry Condition</i></b>	User selects “Save File” from main menu or clicks Save button.
<b><i>Exit Condition</i></b>	File is successfully saved
<b><i>Quality Requirements</i></b>	TBD

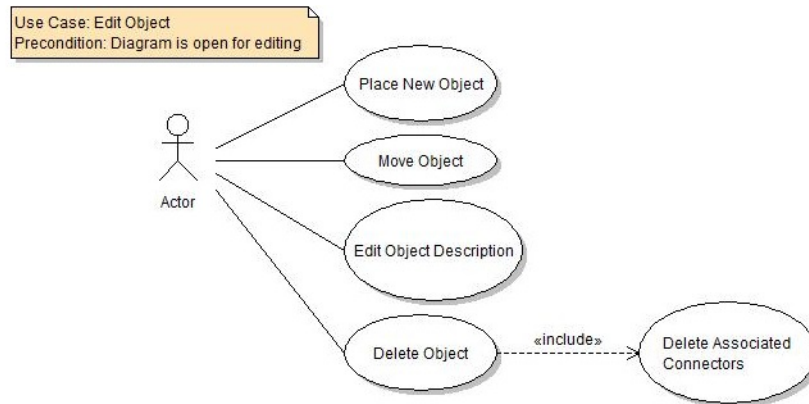
<b><i>Use Case Name</i></b>	<b>Save Diagram As</b>
<b><i>Participating Actor</i></b>	User
<b><i>Flow of Events</i></b>	<ol style="list-style-type: none"> <li>1. User opens File menu and selects “Save As...”</li> </ol> <p>or</p> <ol style="list-style-type: none"> <li>1. User selects “Save” from File menu, but the program has not been previously saved.</li> <li>2. Program displays a dialog box requesting the user to enter a diagram name.</li> <li>3. User enters a diagram name and clicks “Ok.”</li> <li>4. Program saves diagram within existing project.</li> </ol>
<b><i>Entry Condition</i></b>	User clicks “Save As” or clicks “Save” with a previously unsaved diagram.
<b><i>Exit Condition</i></b>	The diagram has been saved under the current working project.
<b><i>Quality Requirements</i></b>	Diagram name may not be the same as another diagram within the project. Diagram may not be saved outside of the project.

<b><i>Use Case Name</i></b>	<b>Print Diagram</b>
<i>Participating Actor</i>	User
<i>Flow of Events</i>	<ol style="list-style-type: none"> <li>1. User initiates print.</li> <li>2. Program sends to printer</li> </ol>
<i>Entry Condition</i>	User initiates Print
<i>Exit Condition</i>	Diagram successfully sent to default printer
<i>Quality Requirements</i>	TBD

<b><i>Use Case Name</i></b>	<b>Close Diagram</b>
<i>Participating Actor</i>	User
<i>Flow of Events</i>	<ol style="list-style-type: none"> <li>1. User clicks the ‘X’ on the diagram tab.</li> <li>2. If the diagram is saved, the program closes the diagram tab.</li> </ol> <p>or</p> <ol style="list-style-type: none"> <li>1. User clicks the ‘X’ on the diagram tab.</li> <li>2. If the program is not saved, the program issues a warning.</li> </ol>
<i>Entry Condition</i>	User clicks the “X” on the diagram tab.
<i>Exit Condition</i>	Diagram tab is closed.
<i>Quality Requirements</i>	Diagram must be saved for the tab to close. Else the user must either choose to save the diagram or discard changes.



**3.2.1.5 Edit Object** These options will be available to the user with a diagram open for modification, who is choosing to take action on an object.



**Figure 2:** Edit an Object Use Case

<i>Use Case Name</i>	<b>Place New Object</b>
<i>Participating Actor</i>	User
<i>Flow of Events</i>	<ol style="list-style-type: none"> <li>1. User selects an object from toolbar.</li> <li>2. User clicks the canvas.</li> <li>3. Program places an object at that location on the canvas.</li> </ol>
<i>Entry Condition</i>	Toolbar is loaded with valid objects for diagram type.
<i>Exit Condition</i>	Object has been successfully placed on drawing canvas.
<i>Quality Requirements</i>	Object may not occupy the same space on the canvas as any other object.

<i>Use Case Name</i>	<b>Edit Object Description</b>
<i>Participating Actor</i>	User
<i>Flow of Events</i>	<ol style="list-style-type: none"> <li>1. User selects an object by clicking on it.</li> <li>2. User right-clicks on the object.</li> <li>3. User selects "Properties" from the drop-down menu.</li> <li>4. User types the desired description in the text box presented.</li> <li>5. User clicks "OK."</li> <li>6. Program displays new description.</li> </ol>
<i>Entry Condition</i>	User clicks on an object.
<i>Exit Condition</i>	Program is now showing all user-made changes.
<i>Quality Requirements</i>	There are limitations as to the number of characters a user may enter in the object description text field.

<i>Use Case Name</i>	<b>Move Object</b>
<i>Participating Actor</i>	User
<i>Flow of Events</i>	<ol style="list-style-type: none"> <li>1. User clicks on object, selecting it.</li> <li>2. User holds down the left mouse button, and moves the mouse to the new location of the object.</li> <li>3. User releases the left mouse button.</li> <li>4. Program draws the object at the new location, with all attached connectors.</li> </ol>
<i>Entry Condition</i>	User selects an object with the left mouse button held down.
<i>Exit Condition</i>	Program has drawn the object at the new location.
<i>Quality Requirements</i>	User may not move an object to a location already occupied by another object.

**3.2.1.6 Edit Connector** These options will be available to the user with a diagram open for modification, who is choosing to take action regarding a connector.

<i>Use Case Name</i>	<b>Place New Connector</b>
<i>Participating Actor</i>	User
<i>Flow of Events</i>	<ol style="list-style-type: none"> <li>1. User clicks on desired connector in toolbar.</li> <li>2. User clicks on the object the connection is coming from, holding down the left mouse button.</li> <li>3. User drags the mouse toward the object to be connected to.</li> <li>4. User releases the mouse button.</li> <li>5. Program draws the connection line between the first and second object.</li> </ol>
<i>Entry Condition</i>	User has selected a connector from the toolbar.
<i>Exit Condition</i>	The connector is successfully drawn between two objects.
<i>Quality Requirements</i>	Only one connector may be placed between any two objects. Some objects may not be able to be connected with certain types of connectors.

<i>Use Case Name</i>	<b>Edit Connector Properties</b>
<i>Participating Actor</i>	User
<i>Flow of Events</i>	<ol style="list-style-type: none"> <li>1. User selects a connector by clicking on it.</li> <li>2. User right-clicks on the connector.</li> <li>3. User selects “Properties” from the drop-down menu.</li> <li>4. User edits description in the text box presented.</li> <li>5. User clicks “OK.”</li> <li>6. Program displays new line description.</li> </ol>
<i>Entry Condition</i>	User right clicks on a connector.
<i>Exit Condition</i>	Program has updated the connector with the new properties.
<i>Quality Requirements</i>	Not all connectors will have the same property options.

<i>Use Case Name</i>	<b>Delete Connector</b>
<i>Participating Actor</i>	User
<i>Flow of Events</i>	<ol style="list-style-type: none"> <li>1. User right clicks on a connector.</li> <li>2. User selects “Delete” from the drop-down menu.</li> <li>3. Program deletes the connector.</li> </ol>
<i>Entry Condition</i>	User right clicks on the connector.
<i>Exit Condition</i>	Connector has been deleted.
<i>Quality Requirements</i>	TBD

### **3.2.2 System feature: Project management tasks suite**

**3.2.2.1 Introduction/Purpose for These Features** Several processes are required to manage projects. These include the saving of files, loading projects from files, and deleting projects.

**3.2.2.2 Input/Output sequence for These Features** The user selects the “File” pull down menu at the top left corner of his or her screen. The options currently supported are “New”, “Open”, “Copy As”, “Save”, “Save As”, and “Delete Project”.

**3.2.2.2.1 New** This creates a new project space.

**3.2.2.2.2 Open** This opens a previously saved project. The user shall be presented with a dialog option to save the previous project. Once completing this dialog, the previous project will be closed and the selected project will be opened.

**3.2.2.2.3 Save** This saves the current state of the project into an XML that can later be used to recreate the project state. If the current project does not have a name, the “Save” option will be an alias for the “Save As” option.

**3.2.2.2.4 Save As** This creates a new project space and then saves the project state into the new project. In contrast with “Copy As”, this option does not affect the previous project.

**3.2.2.2.5 Delete Project** This deletes the current project and all files within the project folder. The user shall be presented with a confirmation dialog before this action is completed. Upon completion, a blank and unnamed project will be active in the project window.

## 4 REQUIREMENTS TRACEABILITY

Feature Name	Req No.	Requirement Description	Priority	SSDD	Test Case(s)	Alpha Release	Beta Release	Final Test
Main Window	2.1.2	All main window features	M		testinstall testlaunch			
Objects	2.1.3	Object properties and functionality	M		testobjects	PASS		
Connectors	2.1.4	Connector properties and functionality	M		testconnectors testselfconnector	FAIL PASS		
Open	2.1.5	Load a pUML file into pUML	M		testsave testsaveas	FAIL		
Save	2.1.6	Save a diagram	M		testsave testsaveas	FAIL PASS		
Diagrams	2.1.7	Diagram type integrity	M		testdiagrams			
Tabs	2.1.8	Tab functionality	M		testtabs			

Priorities are: **Mandatory**, **Low**, **High** SSDD link is version and page number or function name. Test cases and results are file names and **P**ass/**F**ail or % passing.