SYSTEMS AND SOFTWARE REQUIREMENTS SPECIFICATION (SSRS) FOR

Phunctional UML Editor
(pUML)

Version 0.0
February 23, 2012

Prepared for:
Bruce Bolden
and
Dr. Clint Jeffery

Prepared by:
Josh Armstrong
Zach Curtis
Brian Bowles
Logan Evans
Jeremy Klas
Nathan Krussel
Maxine Major
Morgan Weir
David Wells
and
Xiaozhe Shen
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		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		M	Updated SSRS and SSDD	MM
4	02/15/2012		M	Use cases expanded and descriptions added	MM
4	02/23/2012		M	Use case descriptions completed	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 Professor Bruce Bolden 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 0.0.

1.2 PURPOSE

The purpose of the system under development is to create and store UML diagrams. While the system will be used by Computer Science students at the University of Idaho, this document is intended to be read and understood by UICS software designers and coders.

1.3 SCOPE

The pUML software was conceptualized as a Software Engineering class project, and was launched in September 2011 . The pUML project is as of the date of this SSRS publication, incomplete, and has yet no aquirers, users, support agencies at this time. Upon completion, the pUML software will be available only for distribution to the University of Idaho Computer Science department, and will be supported by the development team.

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
DFD	Data Flow Diagram
SDD	Software Design Document, aka SDS, Software Design Specification
SRS	Software Requirements Specification
SSRS	System and Software Requirements Specification

1.5 REFERENCES

There are no references to be cited for the pUML SSRS at this time.

1.6 OVERVIEW AND RESTRICTIONS

This document is for limited release only to UI CS personnel working on the 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.

Sections 5 and up are appendices including original information and communications used to create this document.

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. The program will allow the user to create new diagrams, edit existing diagrams, and save them to access later.

2.3 USER CHARACTERISTICS

The intended user for the pUML software is a software engineer, with a need to organize the parts of the software engineering project. 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 dictated by University of Idaho Computer Science Department faculty, and any further direction this project may take will depend on their decisions. Furthermore, should any decision be made, for example, a new programming language must be utilized, or different features are to be added/removed, this project could change.

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 dependencies, 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. Undo and Redo options will be available to handle errors made on the part of the user. Earlier stored sessions are not a part of the software package at this time, but may be developed at a later release. This software is not designed for any level of flexibility at this time, but a future release may permit integration with other software environments. Testability has not been tested at this time.

2.6.5 Packaging and delivery requirements

The executable system and all associated documentation (i.e., SSRS, SDD, 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 document "drops" will occur 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

No training materials or expectations are tied to this project other than the limited help screens built into the software and the accompanying user manual.

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 are no software application dependencies at this time.

2.6.9 Precedence and criticality of requirements

All requirements have equal weight.

3 SPECIFIC REQUIREMENTS

3.1 EXTERNAL INTERFACE REQUIREMENTS

3.1.1 Hardware Interfaces

- Operating system and environment capable of running QT 4.2.
  - Storage disk

E.g., hard drive, SSD, or secondary flash. 15 MB of space on one of these storage disks will be required to execute the pUML executable. Additional space will be required to save user generated projects.

3.1.2 Software Interfaces

- QT 4.2
  - C++ compiler

Note: Software interfaces are expected to change before the next release.

3.1.3 User Interfaces

- Monitor

Since pUML is a graphical program, a sufficiently large and bright monitor is recommended.

- Keyboard

The user will frequently need to fill in text fields.

- Mouse

The majority of user interaction is through the 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 three 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.

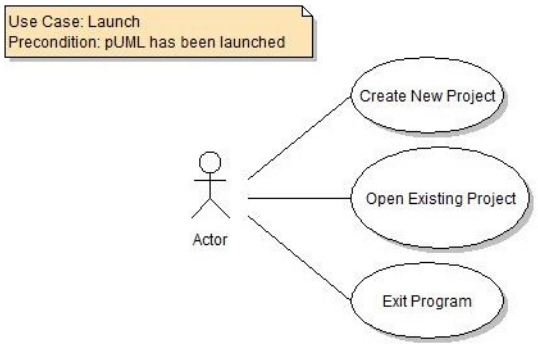

Figure 1: Launch Use Case

Use Case Name	**Create New Project**
Participating Actor	User
Flow of Events	1. User opens File menu and selects "Create New Project" 2. A dialog appears asking the user to enter the project name and save location. 3. User clicks "OK" 4. Program creates a new project for the user at the specified location, and opens the project in pUML.
Entry Condition	User opens File menu and selects "Create New Project."
Exit Condition	Project has been created and is open for modification in pUML.
Quality Requirements	Project name may not be the same as another project at the specified location. Project creation may only occur at program launch, and not while a project has been loaded into pUML.

Use Case Name	**Open Existing Project**
Participating Actor	User
Flow of Events	1. User opens File menu and selects "Open Project." 2. User browses to desired directory and selects an existing pUML project file. 3. The program loads the specified project.
Entry Condition	User opens File menu and selects "Open Project."
Exit Condition	Project has been loaded and is open for modification in pUML.
Quality Requirements	Project must be of pUML file type.

Use Case Name	**Exit Program**
Participating Actor	User
Flow of Events	1. User clicks X. 2. If file is not saved, include "Save File As" use case. 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.

3.2.1.2 New Project

The user has selected to start a new UML project.

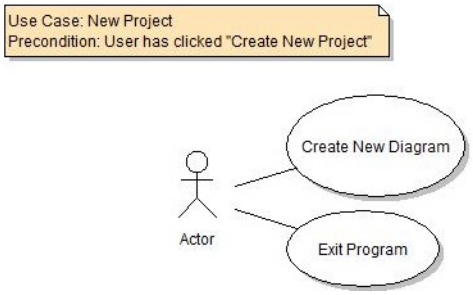

Figure 2: New Project Use Case

Use Case Name	**Create New Diagram**
Participating Actor	User
Flow of Events	1. The user selects new file from menu. 2. Program requests user select a diagram type (includes ChooseDiagramType use case). 3. The program responds by creating a new file with a blank drawing canvas.
Entry Condition	User selects New File from commands
Exit Condition	Program successfully opens new file
Quality Requirements	TBD

Use Case Name	**Choose Diagram Type**
Participating Actor	User
Flow of Events	1. User selects a diagram type. 2. Program loads and displays only objects for the selected diagram type.
Entry Condition	Included from New Diagram use case
Exit Condition	Included from New Diagram use case
Quality Requirements	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.

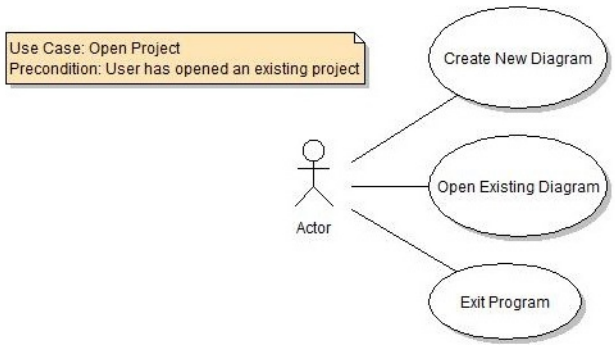

Figure 3: Open Project Use Case

Use Case Name	**Open Existing Diagram**
Participating Actor	User
Flow of Events	1. User selects Open File from menu. 2. Program opens an explorer window. 3. User selects a file. 4. Program loads selected file into program
Entry Condition	User selects Open File from menu.
Exit Condition	File is successfully opened.
Quality Requirements	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.

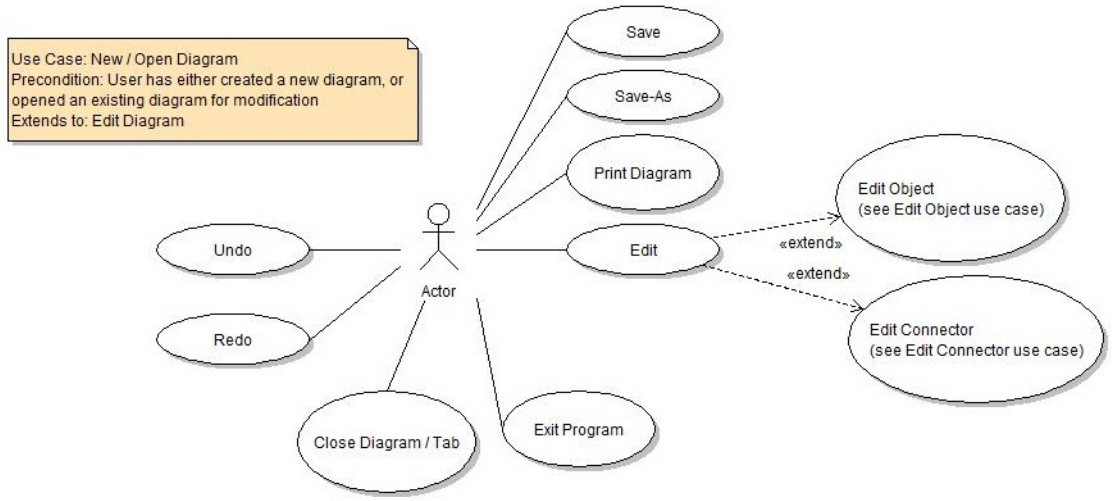

Figure 4: New/Open Diagram Use Case

Use Case Name	**Save File**
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	TBD

Use Case Name	**Save Diagram 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. 3. User enters a diagram name and clicks "Ok." 4. Program saves diagram within existing project.
Entry Condition	User clicks "Save As" or clicks "Save" with a previously unsaved diagram.
Exit Condition	The diagram has been saved under the current working project.
Quality Requirements	Diagram name may not be the same as another diagram within the project. Diagram may not be saved outside of the project.

Use Case Name	**Print Diagram**
Participating Actor	User
Flow of Events	1. User initiates print. 2. Program sends to printer
Entry Condition	User initiates Print
Exit Condition	Diagram successfully sent to default printer
Quality Requirements	TBD

Use Case Name	**Close Diagram**
Participating Actor	User
Flow of Events	1. User clicks the 'X' on the diagram tab. 2. If 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. Else the user must either choose to save the diagram or discard changes.

Use Case Name	**Undo**
Participating Actor	User
Flow of Events	1. User clicks "Undo" 2. Program reverts to the state of the diagram prior to the last change made.
Entry Condition	User clicks "Undo."
Exit Condition	The state of the diagram prior to the most recent change is now the active state.
Quality Requirements	Undo will not work if there are no changes made to the diagram during the current open state.

Use Case Name	**Redo**
Participating Actor	User
Flow of Events	1. User clicks "Redo." 2. Program advances to the state from which the user had previously clicked "Undo."
Entry Condition	User clicks "Redo."
Exit Condition	Current diagram loaded is now a state more current than the one from which the user had clicked "Undo."
Quality Requirements	Redo will not work if there are no changes made to the diagram during the current open state. The user must have previously clicked "Undo" in order for a state to exist to redo.

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.

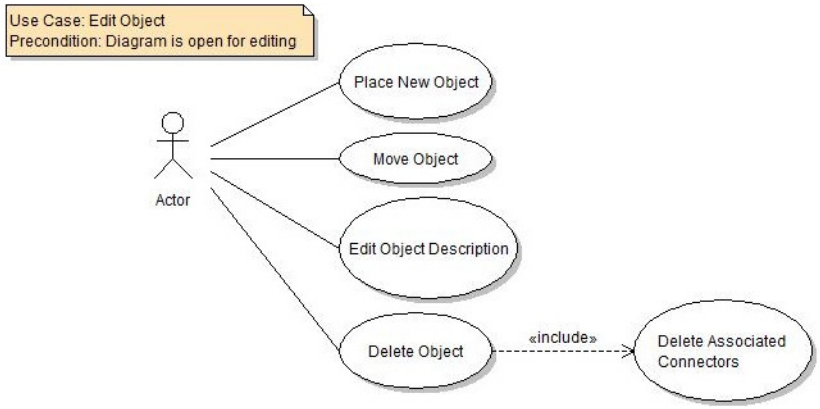

Figure 5: Edit an Object Use Case

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.
Entry Condition	Toolbar is loaded with valid objects for diagram type.
Exit Condition	Object has been successfully placed on drawing canvas.
Quality Requirements	Object may not occupy the same space on the canvas as any other 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 clicks on an object.
Exit Condition	Program is now showing all user-made changes.
Quality Requirements	There are limitations as to the number of characters a user may enter in the object description text field.

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	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.

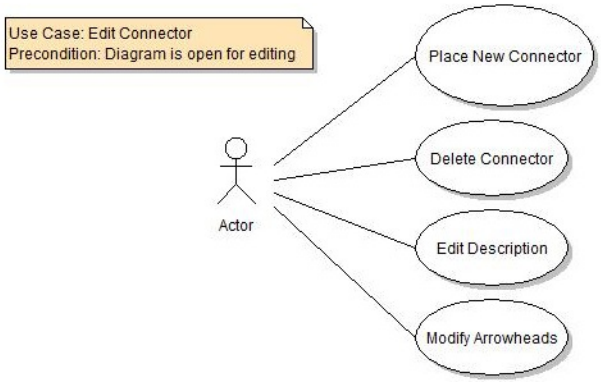

Figure 6: Edit a Connector Use Case

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 object the connection is coming from, holding down the left mouse button. 3. User drags the mouse toward the object to be connected to. 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. Some objects may not be able to be connected with certain types of connectors.

Use Case Name	**Edit Connector Properties**
Participating Actor	User
Flow of Events	1. User selects a connector by clicking on it. 2. User right-clicks on the connector. 3. User selects "Properties" from the drop-down menu. 4. User edits propertes: (a) User types the desired description in the text box presented. (b) User selects desired arrowhead. 5. User clicks "OK." 6. Program displays new description or arrowhead.
Entry Condition	User right clicks on a connector.
Exit Condition	Program has updated the connector with the new properties.
Quality Requirements	Not all connectors will have the same property options.

Use Case Name	**Delete Connector**
Participating Actor	User
Flow of Events	1. User right clicks on a connector. 2. User selects "Delete" from the drop-down menu. 3. Program deletes the connector.
Entry Condition	User right clicks on the connector.
Exit Condition	Connector has been deleted.
Quality Requirements	TBD

3.2.2 System feature: Project management tasks suite

3.2.2.1 Introduction/Purpose of this feature 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 this feature 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", "Revert", "Copy As", "Save", "Save As", and "Delete Project".

3.2.2.2.1 New This creates a new project space. 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 a new project space will be created.

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 Revert This disregards all unsaved changes made to the project and reverts to the saved version. A confirmation dialog shall be presented to the user before this action is completed.

3.2.2.2.4 Copy As This creates a new project space. First, the program will be saved. Then the user shall be presented with a dialog to choose a unique name for the new project space. Once completing this dialog, a copy of the previous project will be cloned into the new project space. The previous project shall be closed and the new project shall be opened.

3.2.2.2.5 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.

3.2.2.2.6 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.

3.2.2.2.7 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

This section shall contain traceability information from each system requirement in this specification to the system (or subsystem, if applicable) requirements it addresses. A tabular form is preferred, but not mandatory.

Feature Name	Req No.	Requirement Description	Priority	SDD	Alpha Release		Beta Release	
					Test Case(s)	Test Res.	Test Case(s)	Test Res.
Select Diagram Type	1.1	Selects the appropriate diagram type	M	N/A	N/A	N/A	N/A	N/A
Save function	2.1	Saves the Diagram to file	M	N/A	N/A	N/A	N/A	N/A
Draw function	2.1	Draws current objects	M	N/A	N/A	N/A	N/A	N/A
Open File	2.1	Opens previously saved file	M	N/A	N/A	N/A	N/A	N/A
New File	2.1	Creates New File	M	N/A	N/A	N/A	N/A	N/A
SSRS and SSDD	2.1	Too much work.	M	N/A	N/A	N/A	N/A	N/A

Priorities are: **M**andatory, **L**ow, **H**igh

SDD link is version and page number or function name.

Test cases and results are file names and **P**ass/**F**ail or % passing.