SYSTEM AND SOFTWARE
DESIGN DESCRIPTION (SSDD) TEMPLATE
(Incorporating Architectural Views and Detailed Design Criteria)
Version A.2, November 2010

FOREWORD

This template was created to provide system and software development projects with a model System and Software Design Description (SSDD) that incorporates both architectural views and detailed design criteria. The template is based on work compiled by Dr. Paul Oman from a large collection of software engineering design document standards discussed in Section 1.5. It has been edited and updated by Dr. Clint Jeffery for use in UI CS 383.

The SSDD template begins on the next page. Just throw away this page and enter your project specifications into the following template. Don't forget to change the headers and footers as necessary. The following conventions are used to guide you in developing your SSDD:

[Text] **Replace** this text with your project design text.

text in italics Notes/instructions to the author. **Delete in your finished document.**

SYSTEM AND SOFTWARE DESIGN DESCRIPTION (SSDD): Incorporating Architectural Views and Detailed Design Criteria
FOR

Phunctional UML Editor (pUML)

Version 0.0
December 10, 2011

Prepared for:
Bruce Bolden

Prepared by:
Josh Armstrong
Zach Curtis
Logan Evans
Jeremy Klas
Maxine Major
Xiaozhe Shen
David Wells
University of Idaho
Moscow, ID 83844-1010

CS383 SSDD
RECORD OF CHANGES (Change History)

Change Number	Date completed	Location of change (e.g., page or figure #)	A M D	Brief description of change	Approved by (initials)	Date approved
df84c744c60f	11/02/11	hg/code/QT Project	A	Added code stubs for node implementation (nodes.cpp, diagrams.cpp, etc)	LE	11/02/11
eae63bed20d2	11/2/11	/.hgignore	A	added .hgignore	LE	11/2/11
b119831d6fea	11/6/11	N/A	M	organized submitted material into folders	LE	11/6/11
82fe1ebeeade	11/6/11	/.hgignore	M	.hgignore to ignore file locks.	LE	11/6/11
6bf8d2e6f033	11/6/11	/hg/documentation/ html	A	Added the html doxygen output to the repository	Auto	11/6/11
db54296b4deb	11/11/11	hg/documentation/ UMLsymbols	A	Updated folder with UML Symbols documentation	MM	11/11/11
1cca683da3f3	11/13/11	/hg/code/mainwindow/	A	Added makefile	LE	11/13/11
4d98e0c6a896	11/13/11	hg/code/mainwindow/	A	submitted UML menu (mainwindow.cpp, GUI.h, etc)	XS	11/13/11
efce0b2f8a24	11/13/11	hg/code/	A	added pUML.cpp	LE	11/13/11
2f45b5a240af	11/17/11	hg/code/Doagram Objects/	A	QT drawing funcitons of circle and actor(circle.cpp...etc)	ZC	11/17/11
8f708577a183	12/2/11	hg/documentation/	A	added UML diagrams and screen shots (use case, interaction, etc)	MM	12/2/2011
ec72fce5eb27	12/4/11	hg/code/mainwindow/	A	Dialog for file->New	DW	12/4/11
442733ec36c2	12/4/11	hg/code/mainwindow/	M	Modified Shen's mainwindow code to gray out tool bars.	DW	12/4/11
8712d7dd4c91	12/4/11	hg/code/mainwindow/ makefile	D	fixed case collision with Makefile and makefile	JA	12/4/11
cc9ebc151eb9	12/5/11	hg/code/mainwindow/	A	Created canvas to allow for drawing area within the main window	JA	12/5/11

6d8c31be300e	12/6/11	/hg/Presentation	A	Submitted User Manual/power point presentation	MM	12/6/11
93098598b83f	12/8/11	hg/documentation/dox	A	created script to run doxygen on all folders inside puml/code	LE	12/8/11

A - ADDED M - MODIFIED D – DELETED

Phunctional UML Editor
TABLE OF CONTENTS

Section Page

1 INTRODUCTION

1.1 IDENTIFICATION

This document has no identification numbers or applicable revisions at this time. All references in this document, excepting items detailed in the change log, may be referenced as revision 0 at this time.

1.2 DOCUMENT PURPOSE, SCOPE, AND INTENDED AUDIENCE

1.2.1 Document Purpose

Phunctional UML Editor software was designed as a graded project for Software Engineering, under the oversight and guidance of Professor Bruce Bolden, at the Unversity of Idaho. This document is required as part of the graded assignment, and provides minimal insight to the design of this incomplete project.

1.2.2 Document Scope and/or Context

This document includes information regarding the design and components of the pUML software.

1.2.3 Intended Audience for Document

This document may be referenced for educational purposes by Computer Science students and faculty at the University of Idaho.

1.3 SYSTEM AND SOFTWARE PURPOSE, SCOPE, AND INTENDED USERS

1.3.1 System and Software Purpose

The pUML software is intended to be a tool utilized by software designers to create UML diagrams.

1.3.2 System and Software Scope/or Context

The pUML software will be designed to provide functionality to create UML diagram projects. Users will be able to create several different types of UML diagrams, create, modify, link, save, and delete objects within individual UML diagrams, and save collections of diagrams stored as part of a project.

1.3.3 Intended Users for the System and Software

The completed product would be available to the general public for purchase.

1.4 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

This section shall list and define all special terms, acronyms and abbreviations used throughout this document. A tabular form is preferable, but not mandatory.

Term or Acronym	Definition
Acquirer	The person, team, or organization that pursues a system or software product or service from a supplier. The acquirer may be a buyer, customer, owner, purchaser, or user. ISO/IEC 42010:2007 (§3.1).

Term or Acronym	Definition
AD	Architectural Description: "A collection of products to document an architecture" ISO/IEC 42010:2007 (§3.4).
Alpha test	Limited release(s) to selected, outside testers
Architect	"The person, team, or organization responsible for systems architecture" ISO/IEC 42010:2007 (§3.2).
Architectural Description	(AD) "A collection of products to document an architecture" ISO/IEC 42010:2007 (§3.4).
Architectural View	"A representation of a whole system from the perspective of a related set of concerns" ISO/IEC 42010:2007 (§3.9).
Architecture	"The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution" ISO/IEC 42010:2007 (§3.5).
Beta test	Limited release(s) to cooperating customers wanting early access to developing systems
Design Entity	"An element (component) of a design that is structurally and functionally distinct from other elements and that is separately named and referenced" IEEE STD 1016-1998 (§3.1).
Design View	"A subset of design entity attribute information that is specifically suited to the needs of a software project activity" IEEE STD 1016-1998 (§3.2).
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
Software Design Description	"A representation of a software system created to facilitate analysis, planning, implementation, and decision making, A blueprint or model of a software system. The SDD is used as the primary medium for communicating software design information" IEEE STD 1016-1998 (§3.4).
SRS	Software Requirements Specification
SSDD	System and Software Design Document
SSRS	System and Software Requirements Specification
System	"A collection of components organized to accomplish a specific function or set of functions" ISO/IEC 42010:2007 (§3.7).
System and Software Architecture and Design Description	An architectural and detailed design description that includes a software system within the context of its enclosing system and describes the enclosing system, the enclosed software, and their relationship and interfaces.
System Stakeholder	"An individual, team, or organization (or classes thereof) with interests in, or concerns, relative to, a system" ISO/IEC 42010:2007 (§3.8).

Term or Acronym	Definition

1.5 DOCUMENT REFERENCES

1) CSDS, *System and Software Requirements Specification Template*, Version 1.0, July 31, 2008, Center for Secure and Dependable Systems, University of Idaho, Moscow, ID, 83844.

2) ISO/IEC/IEEE, *IEEE Std 1471-2000 Systems and software engineering – Recommended practice for architectural description of software intensive systems,* First edition 2007-07-15, International Organization for Standardization and International Electrotechnical Commission, (ISO/IEC), Case postale 56, CH-1211 Genève 20, Switzerland, and The Institute of Electrical and Electronics Engineers, Inc., (IEEE), 445 Hoes Lane, Piscataway, NJ 08854, USA.

3) IEEE, *IEEE Std 1016-1998 Recommended Practice for Software Design Descriptions*, 1998-09-23, The Institute of Electrical and Electronics Engineers, Inc., (IEEE) 445 Hoes Lane, Piscataway, NJ 08854, USA.

4) 3) ISO/IEC/IEEE, *IEEE Std. 15288-2008 Systems and Software Engineering – System life cycle processes,* Second edition 2008-02-01, International Organization for Standardization and International Electrotechnical Commission, (ISO/IEC), Case postale 56, CH-1211 Genève 20, Switzerland, and The Institute of Electrical and Electronics Engineers, Inc., (IEEE), 445 Hoes Lane, Piscataway, NJ 08854, USA.

5) ISO/IEC/IEEE, IEEE Std. 12207-2008, *Systems and software engineering – Software life cycle processes,* Second edition 2008-02-01, International Organization for Standardization and International Electrotechnical Commission, (ISO/IEC), Case postale 56, CH-1211 Genève 20, Switzerland, and The Institute of Electrical and Electronics Engineers, Inc., (IEEE), 445 Hoes Lane, Piscataway, NJ 08854, USA.

1.6 DOCUMENT OVERVIEW

Section 2 of this document describes the system and software constraints imposed by the operational environment, system requirements and user characteristics, and then identifies the system stakeholders and lists describes their concerns and mitigations to those concerns.

Section 3 of this document describes the system and software architecture from several viewpoints, including, but not limited to, the developer's view and the user's view.

Section 4 provides detailed design descriptions for every component defined in the architectural view(s). Sections 5 provides traceability information connecting the original specifications (referenced above) to the architectural components and design entities identified in this document.

Section 6 and beyond are appendices including original information and communications used to create this document.

1.7 DOCUMENT RESTRICTIONS

This document is for LIMITED RELEASE ONLY to UI CS personnel working on the project.

2 CONSTRAINTS and stakeholder concerns

2.1 CONSTRAINTS

2.1.1 Environmental constraints.

The pUML software poses no Environmental constraints at this time .

2.1.2 System requirement constraints.

The pUML software will be designed to function on, at a minimum, Windows 7, Mac OSX, and Linux. Cross platform functionality will minimize portability errors and allow for projects to be migrated between platforms with minimal difficulty.

2.1.3 User characteristic constraints.

University of Idaho Computer Science students and faculty should be able to reasonably understand and operate the pUML software.

2.2 STAKEHOLDER CONCERNS

There are no stakeholders for our software at this time.

3 SYSTEM AND SOFTWARE ARCHITECTURE

3.1 DEVELOPER'S ARCHITECTURAL VIEW

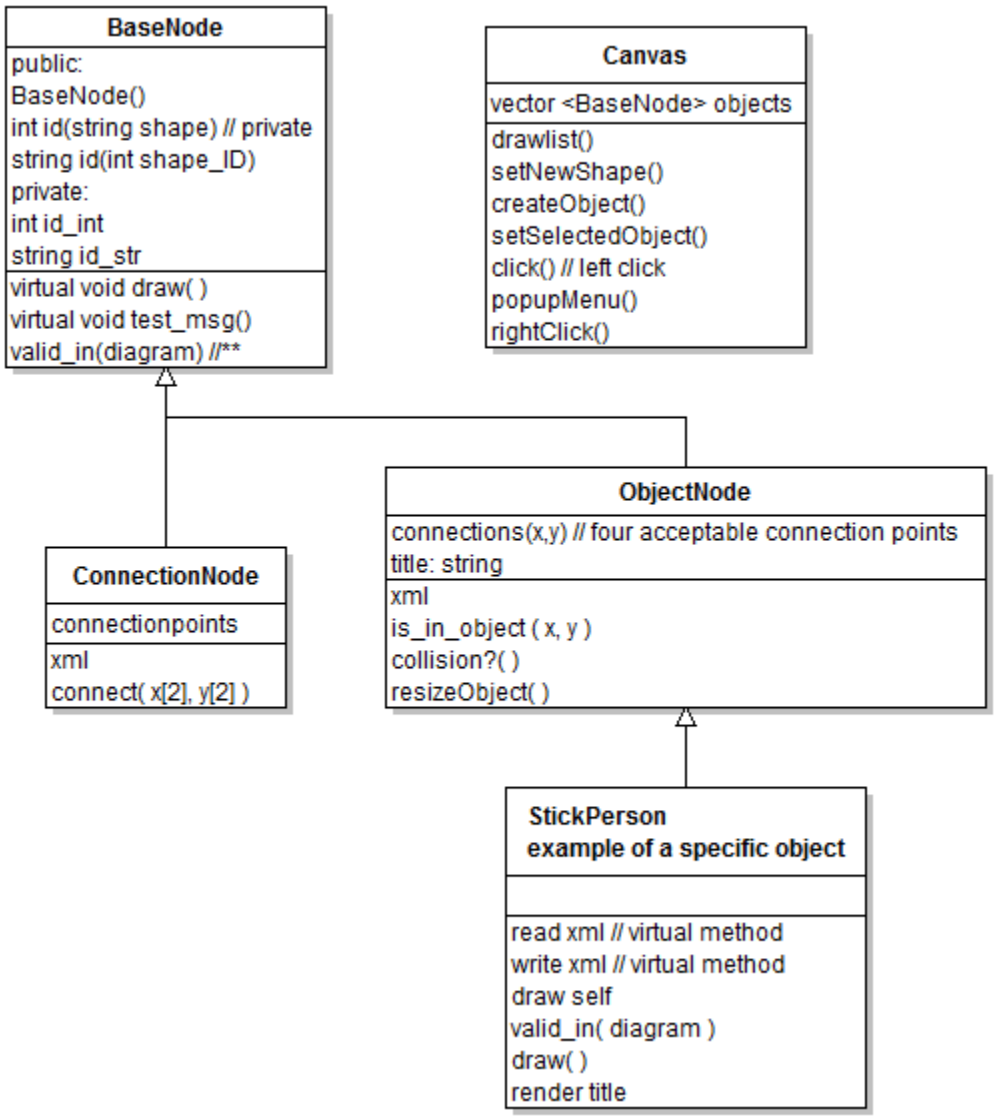
3.1.1 Developer's View Identification

This is the architecture of the program from the viewpoint of the developer. The purpose is to give an overview of the details of the major components of the architecture.

In order to have the program be able to draw diagrams, a custom QWidget is defined called the Canvas. The Canvas holds all the instatiations of nodes and draws them all. It also creates the nodes by handling the mouse click events. The toolbar and menu system lets the Canvas know which type of object will be created next. The Canvas is a member of the MainWindow class, which inherits from QMainWindow.

3.1.2 Developer's View Representation and Description

The Canvas contains a vector container of ObjectNodes. Each ObjectNode has a draw function which takes a QPainter reference as an argument and draws the appropriate figure with the QPainter. The program then defines it's own ObjectNodes, e.g. CircleNode and DiamondNode, and pushes them into the vector. In this way the Canvas can draw each of the nodes in the diagram. To create a new object, it handles a mouse click event and creates a new object of the type specified by a previous call to it's function to set a new object type. The new object is pushed into the vector and then the draw function is called on every node in that vector. When selecting a node to edit or delete, the Canvas takes the X and Y coordinates of the click and translates that into the index of the object selected. Then the Canvas can popup a menu to edit the node or delete the node.



3.1.3 Developer's Architectural Rationale

We decided to create a new QWidget for the canvas so that it can handle click events and have a paint function. We then decided to have the nodes represented by a vector so that it can be easily iterated over and quickly accessed by index. We decided to have the nodes be represented by specific definitions of ObjectNodes so that they can all be pushed into the a vector ObjectNodes. This way each of the nodes can define their own draw function, as well private data such as radius for circles. This allows new objects to be easily created.

3.2 USER'S ARCHITECTURAL VIEW

3.2.1 User's View Identification

This is the viewpoint of the program from the viewpoint of the user. From this viewpoint, there are three major components of the program: the Canvas, the Toolbar and the Menu.

3.2.2 User's View Representation and Description

The menu and the toolbar have redundant functionality. The toolbar has quick access to certain menu items. Either way, the user selects what object he wants to draw and then clicks on the canvas to draw them. Then the user can select objects by clicking on them on the canvas, and right clicking on them to popup a menu. From the popup menu he can delete or edit the object.

3.3 CONSISTENCY OF ARCHITECTURAL VIEWS

There are no known inconsistencies between views.

3.3.1 Detail of Inconsistencies between Architectural Views

NA

3.3.2 Consistency Analysis and Inconsistency Mitigations

NA

4 SOFTWARE DETAILED DESIGN

4.1 DEVELOPER'S VIEWPOINT DETAILED SOFTWARE DESIGN

The Canvas widget is the main viewpoint of our software design.

4.2 COMPONENT/ENTITY DICTIONARY

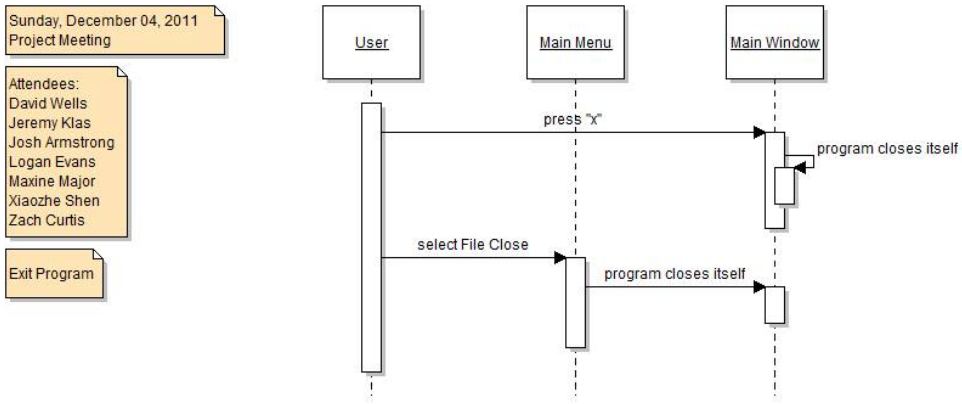
Component/Entity Dictionary				
Name	**Type/Range**	**Purpose/Function**	**Dependencies**	**Subordinates**
Main Window	QMainWindow	To have GUI and house Canvas	N/A	Canvas
Canvas	QWidget	To draw UML diagrams	The Main Window	N/A
Main Menu				
QMenu				
Toolbar				

4.3 COMPONENT/ENTITY DETAILED DESIGN

4.3.1 Detailed Design for Component/Entity: Main Window

4.3.1.1 Introduction/Purpose of this Component/Entity To have a GUI that supports canvas widget.

4.3.1.2 Input for this Component/Entity It should open an XML file.



4.3.1.3 Output for this Component/Entity Should output an XML file to save canvas state.

4.3.1.4 Component/Entity Process to Convert Input to Output Inserting objects into XML tree structure.

4.3.1.5 Design constraints and performance requirements of this Component/Entity Have a functional UI that isn't clunky or hard to use.

4.3.2 Detailed Design for Component/Entity: Canvas

4.3.2.1 Introduction/Purpose of this Component/Entity To draw UML shapes and connectors based on UML diagram type

4.3.2.2 Input for this Component/Entity Mouse click events

4.3.2.3 Output for this Component/Entity Drawing shapes and connectors to screen

4.3.2.4 Component/Entity Process to Convert Input to Output QT drawing/update functions

4.3.2.5 Design constraints and performance requirements of this Component/Entity It should run fast enough that it shouldn't bog down system or run jerky due to too much cpu/gpu usage.

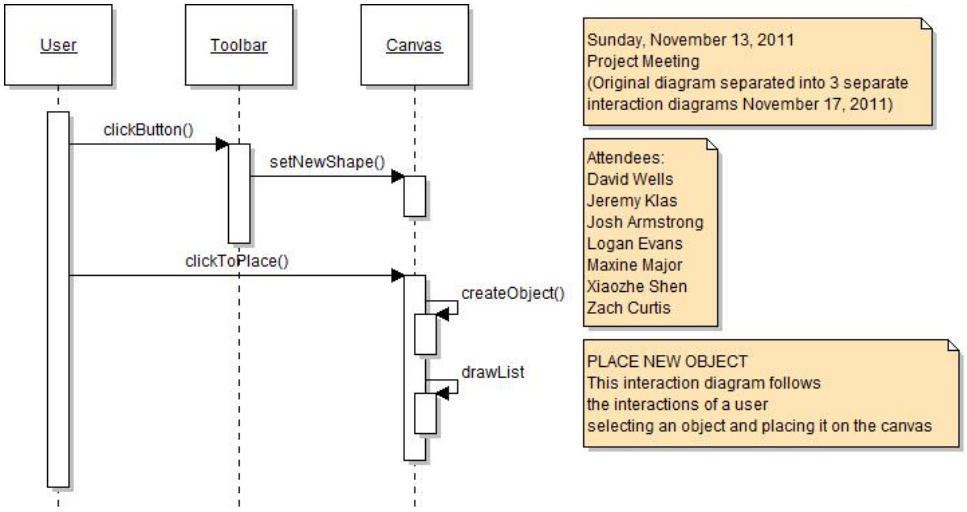
4.3.3 Detailed Design for Component/Entity:

4.3.3.1 Introduction/Purpose of this Component/Entity

4.3.3.2 Input for this Component/Entity

4.3.3.3 Output for this Component/Entity

4.3.3.4 Component/Entity Process to Convert Input to Output



4.3.3.5 Design constraints and performance requirements of this Component/Entity

4.4 DATA DICTIONARY

Data Dictionary				
Name	**Type/Range**	**Defined by...**	**Referenced by...**	**Modified by...**
Main Window	QMainWindow	QWidget	N/A	User
Canvas	QWidget	QWidget	Main Window	User

5 REQUIREMENTS TRACEABILITY

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	Mandatory	N/A	N/A	N/A	N/A	N/A
Save function	2.1	Saves the Diagram to file	Mandatory	N/A	N/A	N/A	N/A	N/A
Draw function	2.1	Draws current objects	Mandatory	N/A	N/A	N/A	N/A	N/A
Open File	2.1	Opens previously saved file	Mandatory	N/A	N/A	N/A	N/A	N/A
New File	2.1	Creates New File	Mandatory	N/A	N/A	N/A	N/A	N/A
SSRS and SSDD	2.1	Too much work.	Mandatory	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.