Advanced Database Design System

[Database Design with Object Constraint Language]

Technical Manual

Geliba

Xinfang Huang

Zijun Liao

Jie Pei

Yaowei Wang

Jie Zhang

Contents

Project Development Target

The new version project includes the design and implement of:

- Compiler of Textual Conceptual Modeling Language(TCML)
- Automatic high quality visualization of textual specification of conceptual schemas.
- Automatic transformation of textual specification of conceptual schemas into relational schemas, DTD, and XML schemas.
- Error messaging and quality evaluation subsystem.
- Syntax driven editor for specification and implementation of conceptual schemas

The group of students will obtain a formal specification of Textual Conceptual Modeling Language (TCML) from a project supervisor who will also play a role of a client for the project.

Environment Setup

Introduction

This document shows how to set the environment for our project based on Windows 7 32-bits operating system(All screen shots are from a Win7 32bits system). It has also included the alternative path options and setting steps, while dealing with different Windows operating systems.

Requirments

Software	Download link
NetBeans IDE 7.2	netbeans-7.2-ml-windows.exe
Qt Libraries 4.8.4 for Windows(MinGW)	qt-win-opensource-4.8.4-mingw.exe
Qt Creator 2.7.0 for Windows	qt-creator-windows-opensource-2.7.0.exe
Minimalist GNU for Windows(MinGW)	mingw-get-inst-20120426.exe
MSYS 1.0.10	MSYS-1.0.10.exe
Flex for Windows	flex-2.5.4a-1.exe
Bison for Windows	bison-2.4.1-setup.exe
OpenGL Utility Toolkit	glut-3.7.6-bin.zip

*Remark: if any of these softwares can not be opened by clicking given links, please refer to Links.

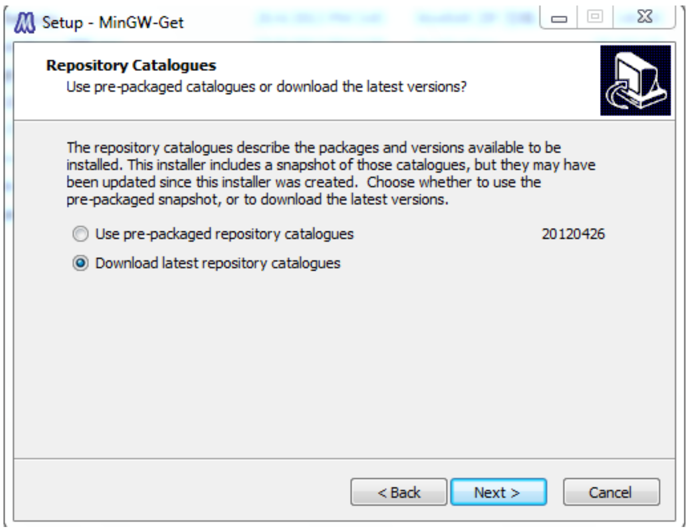
All softwares are installed by default directories(except Flex and Bison, see the installation) or in C drive(system drive), and all settings are base on this. For custom installations, please aware the changes to installation directories.

NetBeans Installation

Install NetBeans IDE 7.2 (only C/C++ plugin is need for the project, be aware of that main components must be installed).

MinGW Installation

For installing MinGW, network connection should be available on your computer and it is strongly recommended that do not install MinGW in any location with spaces in the path name. During installation, select "**Download latest repository catalogues**" and continue. It will dowload the latest virsion.



Setup - MinGW-Get

Repository Catalogues

Use pre-packaged catalogues or download the latest versions?

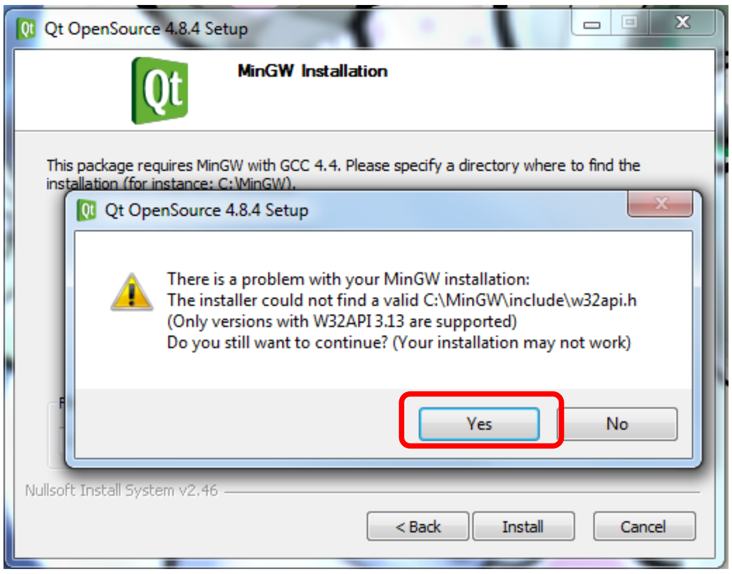
The repository catalogues describe the packages and versions available to be installed. This installer includes a snapshot of those catalogues, but they may have been updated since this installer was created. Choose whether to use the pre-packaged snapshot, or to download the latest versions.

○ Use pre-packaged repository catalogues 20120426

● Download latest repository catalogues

Qt Installation

Installations of Qt libraries should be done by simply following the installing instructions. The following will be show during install.



Qt OpenSource 4.8.4 Setup

MinGW Installation

This package requires MinGW with GCC 4.4. Please specify a directory where to find the installation (for instance: C:\MinGW).

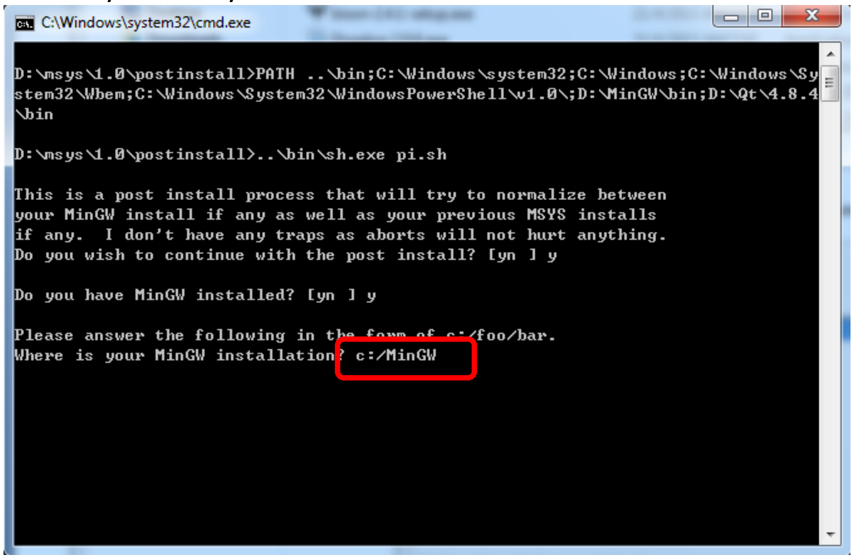
Qt OpenSource 4.8.4 Setup

There is a problem with your MinGW installation:
The installer could not find a valid C:\MinGW\include\w32api.h
(Only versions with W32API 3.13 are supported)
Do you still want to continue? (Your installation may not work)

Check the file "w32api.h" is under the folder "MinGW\include", and just click "Yes" to continue the installation. This is because the QT library can't find the file "w32api.h", this will be happen when the MinGW version is newer then version 4.4. Installations of Qt creator should also follow the installing instructions.

MSYS Installation

Install MSYS 1.0.10. At the end of installation, there should be a pop up window for a post install process. To the questions in the window, type "**y**"s to continue. Specify the path where MinGW installed, when it is requested. By default, the path should be "**C:/MinGW**" (**Note: it is not back slash "\\"**). Please reinstall MSYS if you didn't specify the MinGW directory correctly


C:\Windows\system32\cmd.exe

```
D:\msys\1.0\postinstall>PATH ..\bin;C:\Windows\system32;C:\Windows;C:\Windows\Sy
stem32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;D:\MinGW\bin;D:\Qt\4.8.4
\bin

D:\msys\1.0\postinstall>..\bin\sh.exe pi.sh

This is a post install process that will try to normalize between
your MinGW install if any as well as your previous MSYS installs
if any.  I don't have any traps as aborts will not hurt anything.
Do you wish to continue with the post install? [yn ] y

Do you have MinGW installed? [yn ] y

Please answer the following in the form of c:/foo/bar.
Where is your MinGW installation? c:/MinGW
```

Flex and Bison Installation

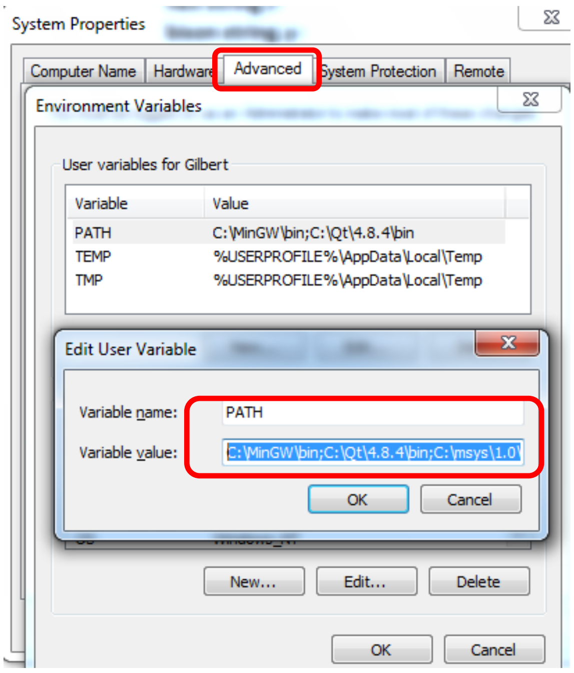
Make a full installation for Flex and Bison in "**C:\GnuWin32**"(**Note: without spaces in the name**). Do not install it in the default directory because Bison has problems with spaces in the directory name. For running Flex and Bison, Environment variable has to be set to include the bin directory. For testing: create two files "hellow.l" and "hellow.y" (new Text Document, then change the file extention).

Environment Variables Settings

For **Windows 7**, right click **Computer -> Properties -> Advanced system settings -> Environment Variables**.

For **Windows XP**, right **click My Computer -> Properties -> Advanced tab -> Environment Variables**

Under **User variables**, create a new variable called "**PATH**", and for **Variable value**, type **"C:\MinGW\bin;C:\Qt\4.8.4\bin;C:\msys\1.0\bin;C:\GnuWin32\bin".**



User variables for Gilbert

Variable	Value
PATH	C:\MinGW\bin;C:\Qt\4.8.4\bin
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

Variable name: PATH

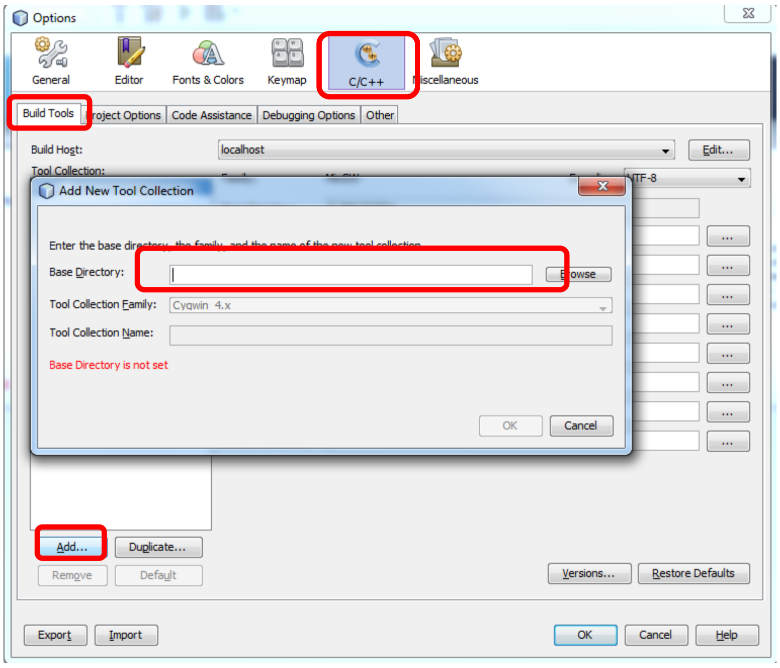
Variable value: C:\MinGW\bin;C:\Qt\4.8.4\bin;C:\msys\1.0\

Press **OK** to close all windows for environment variables setting windows.

NetBeans Settings

This part also includes how to set including directories for Qt in NetBeans.

Run NetBeans IDE, click **Tools -> Options** -> **C/C++** -> **Build Tools** -> **Add...**
In **Add New Tool Collection window**, press **Browse** -> direct to **"C:\MinGW\bin"** -> **OK**.



Build Host: localhost

Tool Collection:

Enter the base directory, the family, and the name of the new tool collection.

Base Directory:

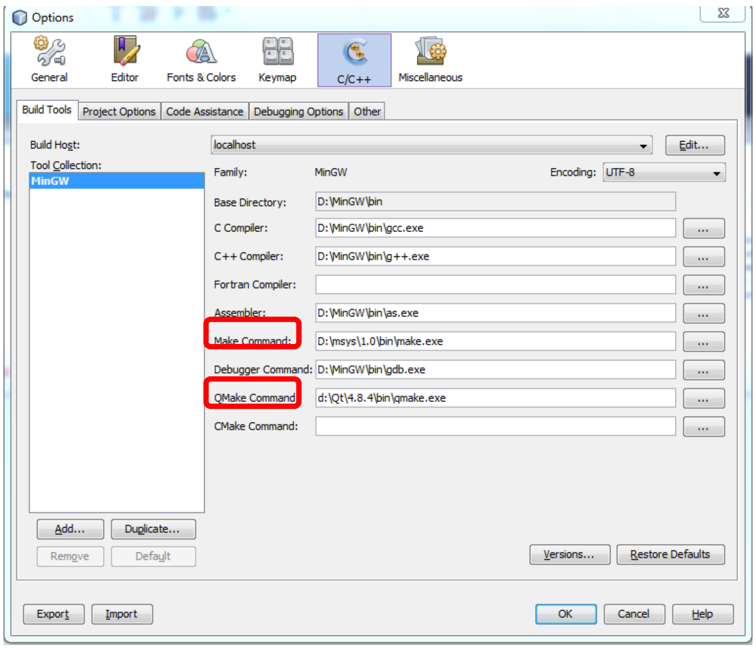
Tool Collection Family: Cygwin 4.x

Tool Collection Name:

Base Directory is not set

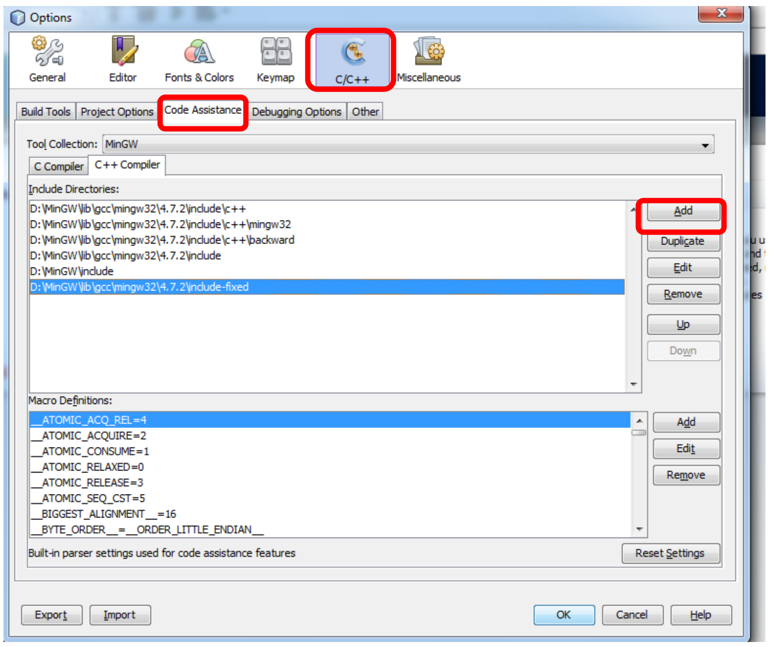
MinGW should be automatically detected, then press **OK**. If "**Make Command**:" and "**Qmake Command**:" are not set, you have to do it manually, which are

"C:\msys\1.0\bin\make.exe" and "C:\Qt\4.8.4\bin\qmake.exe" respectivly. Do not close the setting window yet.



Build Host: localhost
Tool Collection: MinGW
Family: MinGW
Encoding: UTF-8
Base Directory: D:\MinGW\bin
C Compiler: D:\MinGW\bin\gcc.exe
C++ Compiler: D:\MinGW\bin\g++.exe
Fortran Compiler:
Assembler: D:\MinGW\bin\as.exe
Make Command: D:\msys\1.0\bin\make.exe
Debugger Command: D:\MinGW\bin\gdb.exe
QMake Command: d:\Qt\4.8.4\bin\qmake.exe
CMake Command:

Press **Code Assistance tab** -> **C++ Compiler** -> **Add** all directories under **"C:\Qt\4.8.4\include"** (Now for out project, **"C:\Qt\4.8.4\include\QtGui"**,**"C:\Qt\4.8.4\include"**, **"C:\Qt\4.8.4\include\QtCore"**, and **"C:\Qt\4.8.4\include\Qt"** are needed). Press **OK** to close all setting windows.



Tool Collection: MinGW

Include Directories:

```
D:\MinGW\lib\gcc\mingw32\4.7.2\include\c++
D:\MinGW\lib\gcc\mingw32\4.7.2\include\c++\mingw32
D:\MinGW\lib\gcc\mingw32\4.7.2\include\c++\backward
D:\MinGW\lib\gcc\mingw32\4.7.2\include
D:\MinGW\include
D:\MinGW\lib\gcc\mingw32\4.7.2\include-fixed
```

Macro Definitions:

```
__ATOMIC_ACQ_REL=4
__ATOMIC_ACQUIRE=2
__ATOMIC_CONSUME=1
__ATOMIC_RELAXED=0
__ATOMIC_RELEASE=3
__ATOMIC_SEQ_CST=5
__BIGGEST_ALIGNMENT__=16
__BYTE_ORDER__=__ORDER_LITTLE_ENDIAN__
```

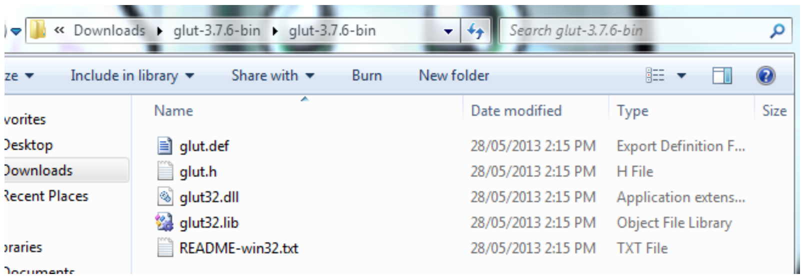
Built-in parser settings used for code assistance features

There could be unresolved identifier (red underline) when using Qt libraries, so you may need to restart your computer. This should solve the problem

OpenGL Utility Toolkit Installation

For this part, you should have the zip file downloaded (see Requirments). Installation for OpenGL is all done manually.

Unzip **glut-3.7.6-bin.zip**(download from the link webpage at end of the document), and there should be five files in the zip file.



Name	Date modified	Type	Size
glut.def	28/05/2013 2:15 PM	Export Definition F...	
glut.h	28/05/2013 2:15 PM	H File	
glut32.dll	28/05/2013 2:15 PM	Application extens...	
glut32.lib	28/05/2013 2:15 PM	Object File Library	
README-win32.txt	28/05/2013 2:15 PM	TXT File	

Copy **glut.h** to the MinGW\include\GL directory.

Copy **glut32.lib** to your build directory (i.e., the directory that you compile into and link from).

Copy **glut32.dll** to the same directory where your executable will be created.

(You can actually put glut32.dll in any directory in your path.)

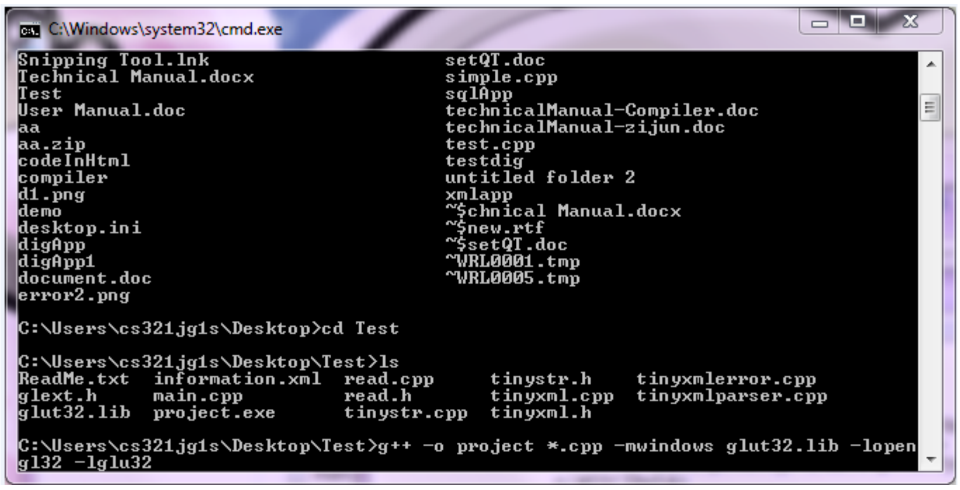
For now, OpenGL Utility Toolkit Installation has been done.

OpenGL Compile with Command Line

Start the command line board and enter the directory where your executable OpenGL file located, then compile with the command line below:

g++ -o project *.cpp -mwindows glut32.lib -lopengl32 -lglu32

When you link, you must link-in glut32.lib (and not use the -lglut32).



```
Snipping Tool.lnk                      setQT.doc
Technical Manual.docx                  simple.cpp
Test                                   sqlApp
User Manual.doc                        technicalManual-Compiler.doc
aa                                     technicalManual-zijun.doc
aa.zip                                 test.cpp
codeInHtml                             testdig
compiler                               untitled folder 2
d1.png                                 xmlapp
demo                                   ~$chnical Manual.docx
desktop.ini                            ~$new.rtf
digApp                                 ~$setQT.doc
digApp1                                ~WRL0001.tmp
document.doc                           ~WRL0005.tmp
error2.png

C:\Users\cs321jg1s\Desktop>cd Test

C:\Users\cs321jg1s\Desktop\Test>ls
ReadMe.txt   information.xml  read.cpp     tinystr.h    tinyxmlerror.cpp
glext.h      main.cpp         read.h       tinyxml.cpp  tinyxmlparser.cpp
glut32.lib   project.exe      tinystr.cpp  tinyxml.h

C:\Users\cs321jg1s\Desktop\Test>g++ -o project *.cpp -mwindows glut32.lib -lopen
gl32 -lglu32
```

Links

NetBeans IDE 7.2:
All versions of NetBeans IDE can be found through following web site.
https://netbeans.org/downloads/

Qt Libraries 4.8.4 for Windows and Qt Creator 2.7.0 for Windows:
Dowload link can be found in the following web site.
http://qt-project.org/downloads/

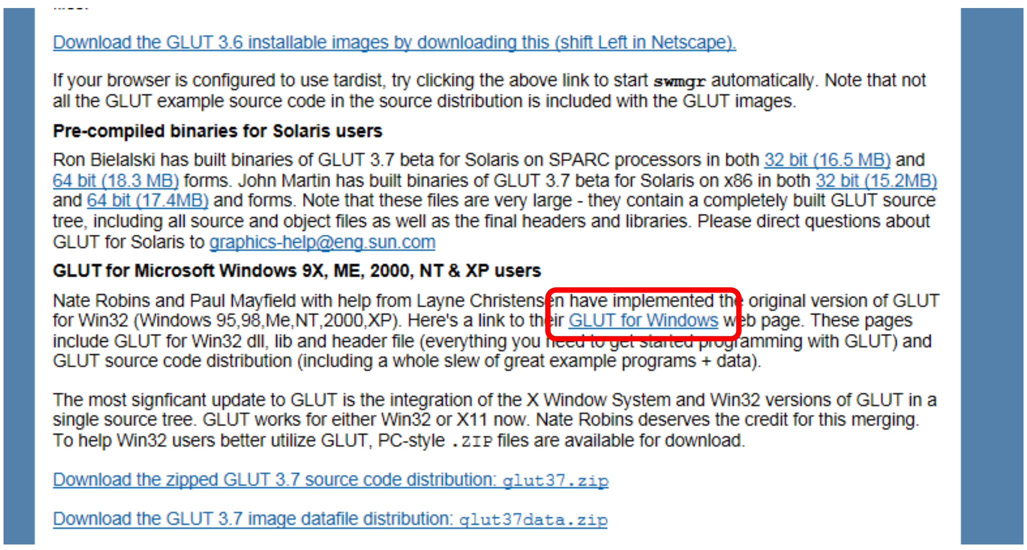
MinGW:
Download link can be found in the following web page.
http://www.mingw.org/wiki/InstallationHOWTOforMinGW

MSYS 1.0.10:
Download link can be found in the following web page.
http://www.mingw.org/wiki/MSYS

Flex and Bison for Windows:

Flex:
Download link can be found in the following web page.
http://gnuwin32.sourceforge.net/packages/flex.htm
Bison:
Download link can be found in the following web page.
http://gnuwin32.sourceforge.net/packages/bison.htm

OpenGL Utility Toolkit:
Glut-3.7.6-bin.zip
Download link can be found in the web page shows below:
http://www.opengl.org/resources/libraries/glut/glut_downloads.php



Download the GLUT 3.6 installable images by downloading this (shift Left in Netscape).

If your browser is configured to use tardist, try clicking the above link to start **swmgr** automatically. Note that not all the GLUT example source code in the source distribution is included with the GLUT images.

Pre-compiled binaries for Solaris users

Ron Bielalski has built binaries of GLUT 3.7 beta for Solaris on SPARC processors in both 32 bit (16.5 MB) and 64 bit (18.3 MB) forms. John Martin has built binaries of GLUT 3.7 beta for Solaris on x86 in both 32 bit (15.2MB) and 64 bit (17.4MB) and forms. Note that these files are very large - they contain a completely built GLUT source tree, including all source and object files as well as the final headers and libraries. Please direct questions about GLUT for Solaris to graphics-help@eng.sun.com

GLUT for Microsoft Windows 9X, ME, 2000, NT & XP users

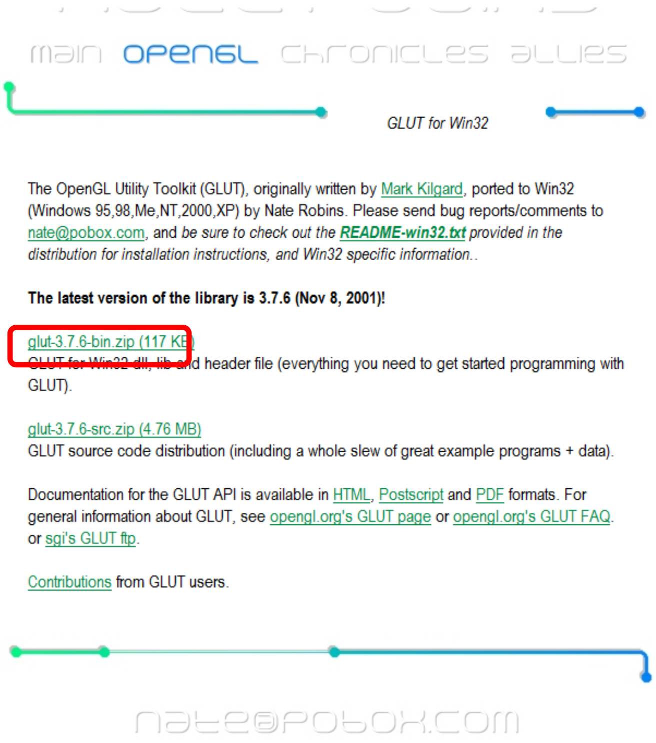
Nate Robins and Paul Mayfield with help from Layne Christensen have implemented the original version of GLUT for Win32 (Windows 95,98,Me,NT,2000,XP). Here's a link to their GLUT for Windows web page. These pages include GLUT for Win32 dll, lib and header file (everything you need to get started programming with GLUT) and GLUT source code distribution (including a whole slew of great example programs + data).

The most signficant update to GLUT is the integration of the X Window System and Win32 versions of GLUT in a single source tree. GLUT works for either Win32 or X11 now. Nate Robins deserves the credit for this merging. To help Win32 users better utilize GLUT, PC-style `.ZIP` files are available for download.

Download the zipped GLUT 3.7 source code distribution: `glut37.zip`

Download the GLUT 3.7 image datafile distribution: `glut37data.zip`

Then the webpage will jump to another link webpage and choose the file in the red block:



GLUT for Win32

The OpenGL Utility Toolkit (GLUT), originally written by Mark Kilgard, ported to Win32 (Windows 95,98,Me,NT,2000,XP) by Nate Robins. Please send bug reports/comments to nate@pobox.com, and *be sure to check out the* ***README-win32.txt*** *provided in the distribution for installation instructions, and Win32 specific information.*.

The latest version of the library is 3.7.6 (Nov 8, 2001)!

glut-3.7.6-bin.zip (117 KB)
GLUT for Win32 dll, lib and header file (everything you need to get started programming with GLUT).

glut-3.7.6-src.zip (4.76 MB)
GLUT source code distribution (including a whole slew of great example programs + data).

Documentation for the GLUT API is available in HTML, Postscript and PDF formats. For general information about GLUT, see opengl.org's GLUT page or opengl.org's GLUT FAQ. or sgi's GLUT ftp.

Contributions from GLUT users.

System Build

Before start building system, the environment should be set. To see how to set the environment, please check the document "Project Environment setup.pdf".

== Overview ==

Our system consists of seven executable files. They should be placed in a single folder to construct the system. The name of the folder is arbitrary, but the names of the executables are restricted except one for GUI(Qt). First of all, create an empty folder somewhere you prefer (eg. create a folder named "System" on the desktop). After you have got all executables needed, place them in the folder you have created.

== Producing executables ==

=> project.exe <=

Execute with NetBeans, open a project and direct to the directory where the project is.
The project and its source files can be found under the subdirectory "projectGUI". Build and run the project
The executable file "project.exe" can be found in the subdirectory "projectGUI\dist\Release\MinGW-Windows".
You can place the produced executable into the system directory by double clicking "CopyGUI.bat".

== Build system ==

Double click "BuildSystem.bat" to build the system. It will create a folder called "csci321ADDS" in current directory and copy all executables and DLLs into the folder.

* There might be a problem when building "project.exe" using the script "BuildSystem.bat".

It may be because of incorrect setup for Qt. If the script not works on your computer, please use NetBeans to build "project.exe". Details can be found above.

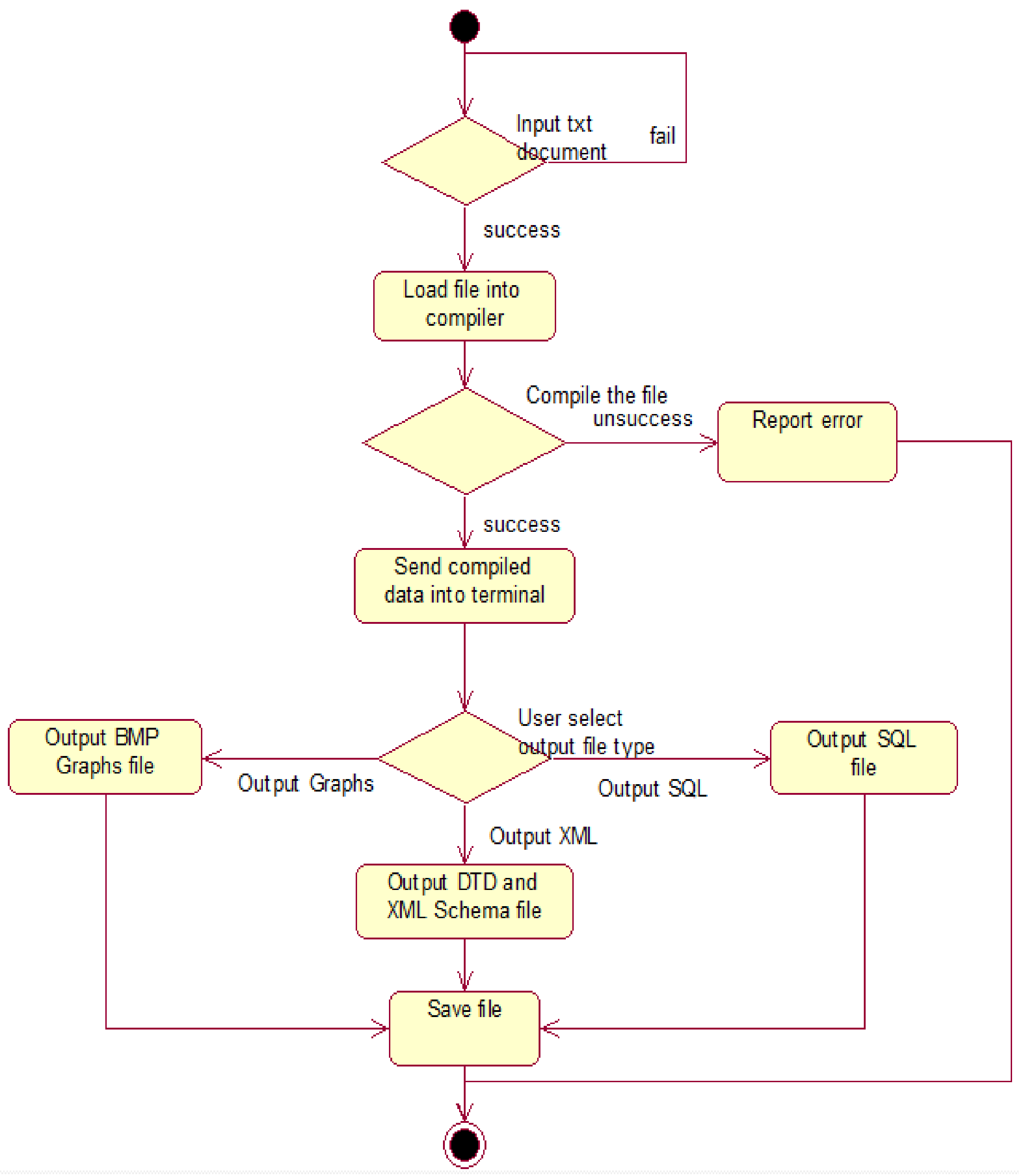
System will graphically look like as shown below(Note: names of Main folder arbitrary):

```
 -<Main folder>--------------------
|file                              |
|compiler.exe                      |
|dtdapp.exe                        |
|digApp.exe                        |
|project.exe                       |
|internalSystem.exe                |
|sqlapp.exe                        |
|xmlapp.exe                        |
 ----------------------------------------

 -<file>-------------------------------
|help.txt                          |
|ProjectCreator.txt                |
----------------------------------------
```

System Implement

Logical

Here is a flowchart which represents the overall flow of logic during software running:



Load Resource File

Basic Flow:

User Action	System Responds
User inputs a txt document.	Checking the file path, displaying a button which allows user to compile the file.

Exception Flow:

User Action	System Responds
The input file is not exit	If fail ask user input file again. If success, displaying a button which allows user to compile the file.

Compile Resource File

Basic Flow:

User Action	System Responds
User compiles the file.	Loading file into the compiler, displaying a checkbox which allow user to choose which kinds of file type to output.

Exception Flow:

User Action	System Responds
The compile process has error or compile failed.	Loading file into the compiler. If compiling the file unsuccessful, report error and exit.

Result Scheme Generate

Basic Flow:

User Action	System Responds
User select the checkbox, there are three selections can be chosen. Graph, SQL and XML.	Outputting the file type which user choose. Displaying the path to let user choose where these files will be saved.

Save Result Files

Basic Flow:

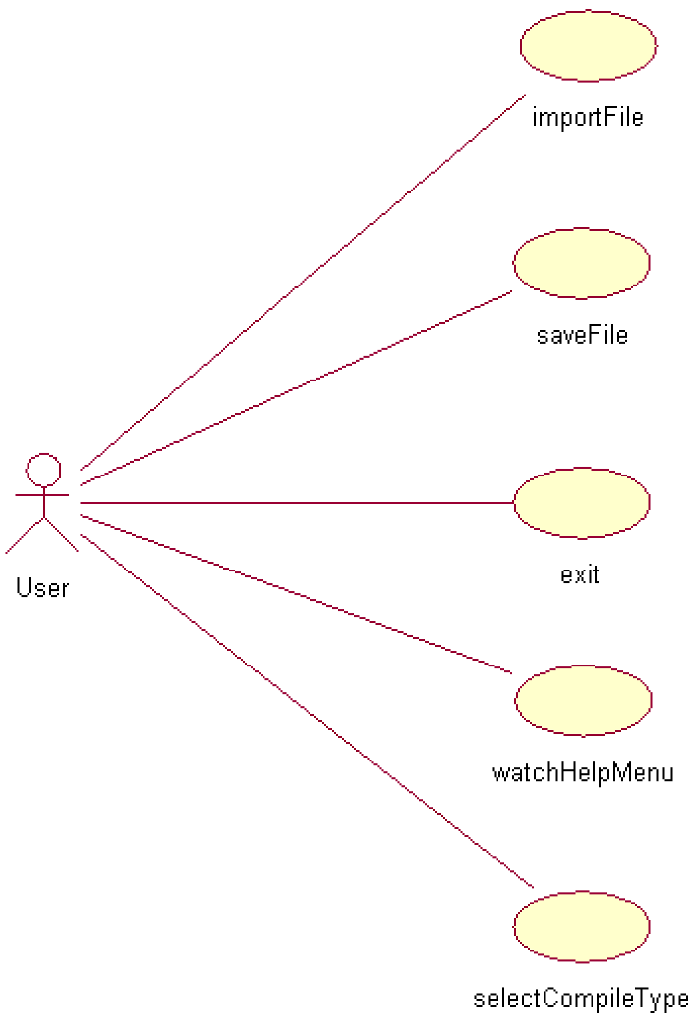
User Action	System Responds
User choose the file path and pressing “save” button	Saving these file which user ask to output.

Exception Flow:

User Action	System Responds
User chooses the file path and pressing “save” button as default type.	Saving these file as default path and default name
User chooses the file path and pressing “save” button as customize type	Saving these file which ask user enter the filename and save path

User Case Diagram

Interface



Use Case Description

Use Case: ImportFile
Primary Actor: User
Secondary Actor: -
Preconditions: -
Success End Condition: Import file success.
Failed End Condition: cannot find file.
Trigger: the user uses the system
BASIC FLOW 1. The system displays the import file button 2. User clicks import file button. 3. The system will check the file path and save it.
ALENTERNATIVE FLOW 3a.The system cannot find the file and display error message. 3a.1 The system will tell user re-enter filename or exit.

Use Case: saveFile
Primary Actor: User
Secondary Actor: -
Preconditions: -
Success End Condition: save success.
Failed End Condition: compile failed or cannot open file.
Trigger: the user uses the system
BASIC FLOW 1. The system displays the import file button 2. User clicks import file button. 3. The system will check the file path and save it. 4. The system will let user to select compile type. 5. The system will call compiler to compile the file. 6. The system will ask user whether the file need to be saved and enter the name. 7. The system will save file.
ALENTERNATIVE FLOW 3a.The system cannot find the file and display error message. 3a.1 The system will tell user re-enter filename or exit. 5a. The file is not correct and the system will send an error message.

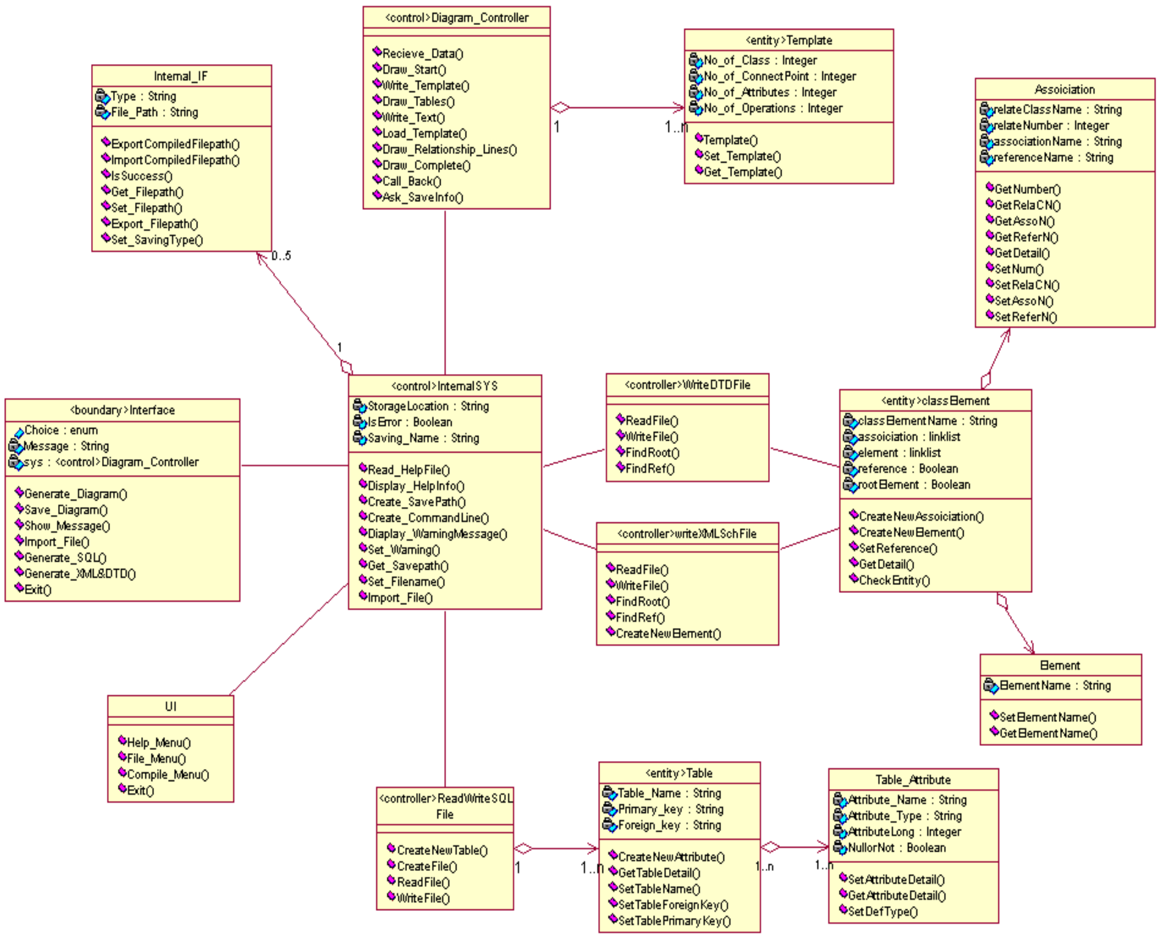
Use Case: slesctCompileType
Primary Actor: User
Secondary Actor: -
Preconditions: -
Success End Condition: the system will compile the file.
Failed End Condition: cannot open file.
Trigger: the user uses the system
BASIC FLOW
1. The system displays the import file button
2. The user clicks import file button.
3. The system will check the file path and save it.
4. The system will let user to select compile type.
ALENTERNATIVE FLOW
3a.The system cannot find the file and display error message.
3a.1 The system will tell user reenter filename or exit.

Use Case: exit
Primary Actor: User
Secondary Actor: -
Preconditions: -
Success End Condition: system close
Failed End Condition: -
Trigger: the user uses the system
BASIC FLOW
1. The system displays the exit file button
2. User clicks exit file button.
3. The system will close.
ALENTERNATIVE FLOW

Use Case: watchHelpMenu
Primary Actor: User
Secondary Actor: -
Preconditions: -
Success End Condition: Display help information.
Failed End Condition: Cannot find help.txt file
Trigger: the user uses the system
BASIC FLOW
1. The system displays the Help Menu button
2. The user clicks Help Menu button.
3. The system will read help.txt file.

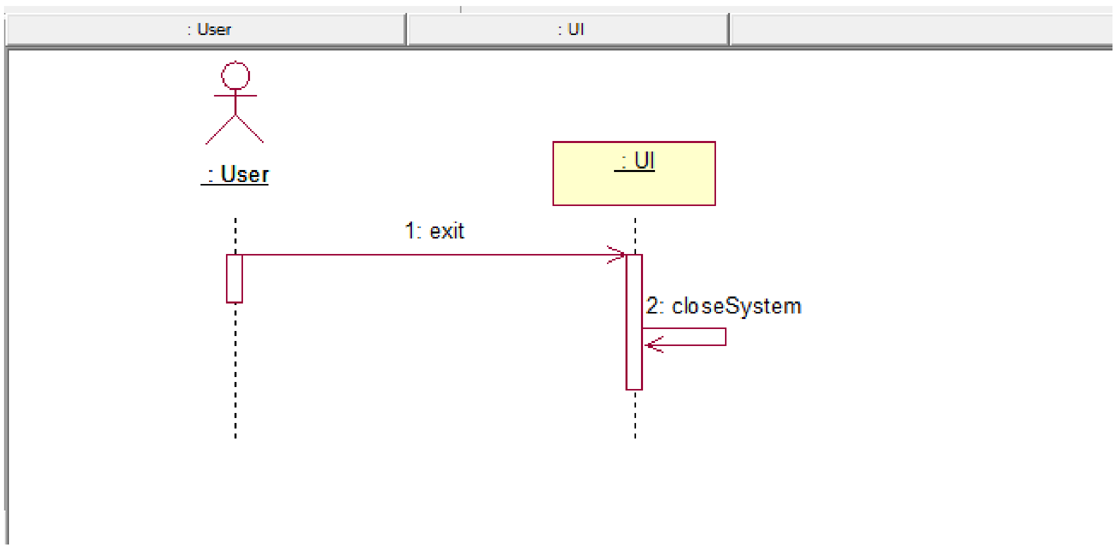
4. The system will display the help information.
ALENTERNATIVE FLOW 3a.The system cannot find the file and display error message.

Main System Class

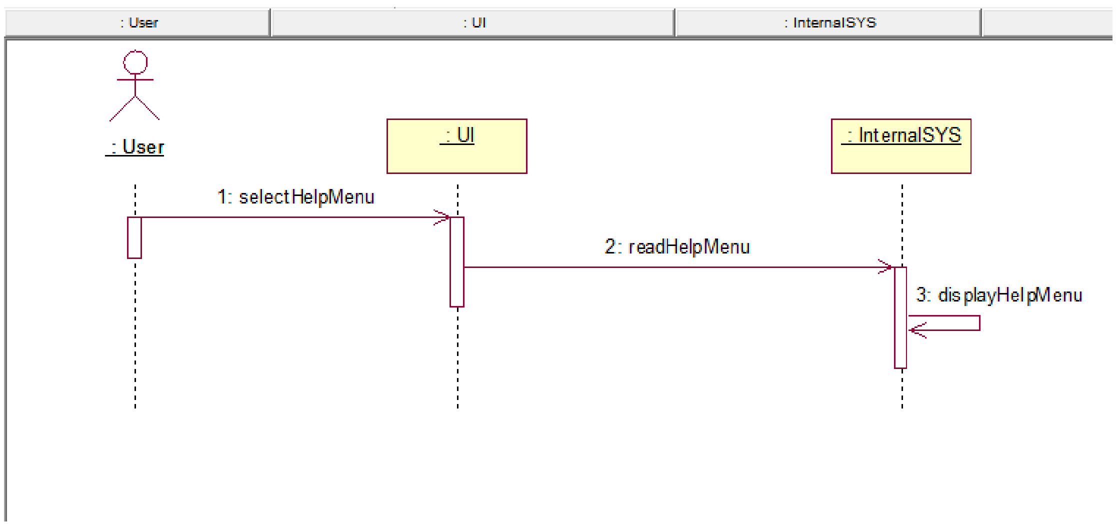


Internal System

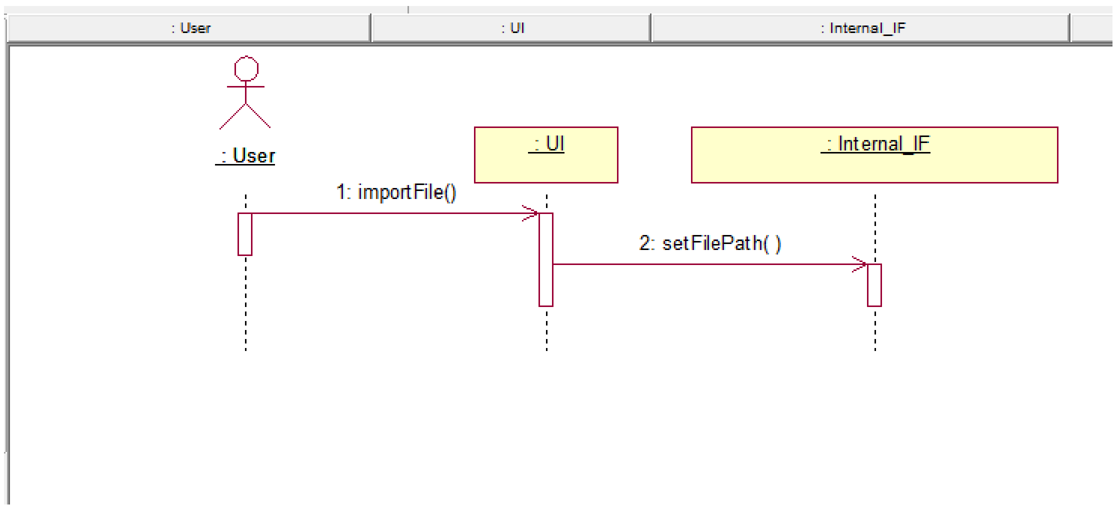
Exit System



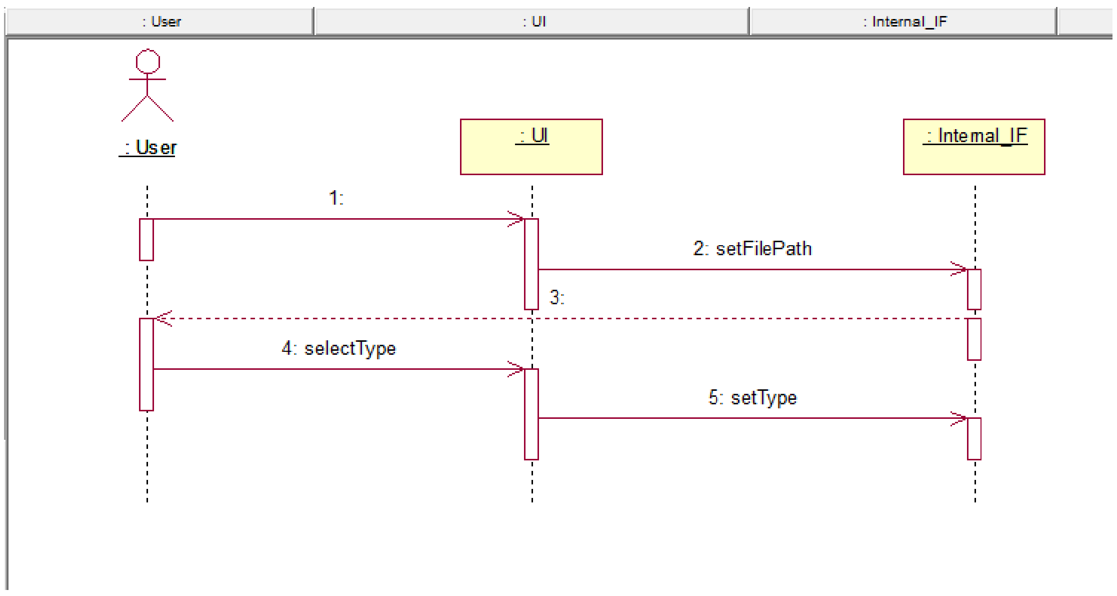
Display Help Information



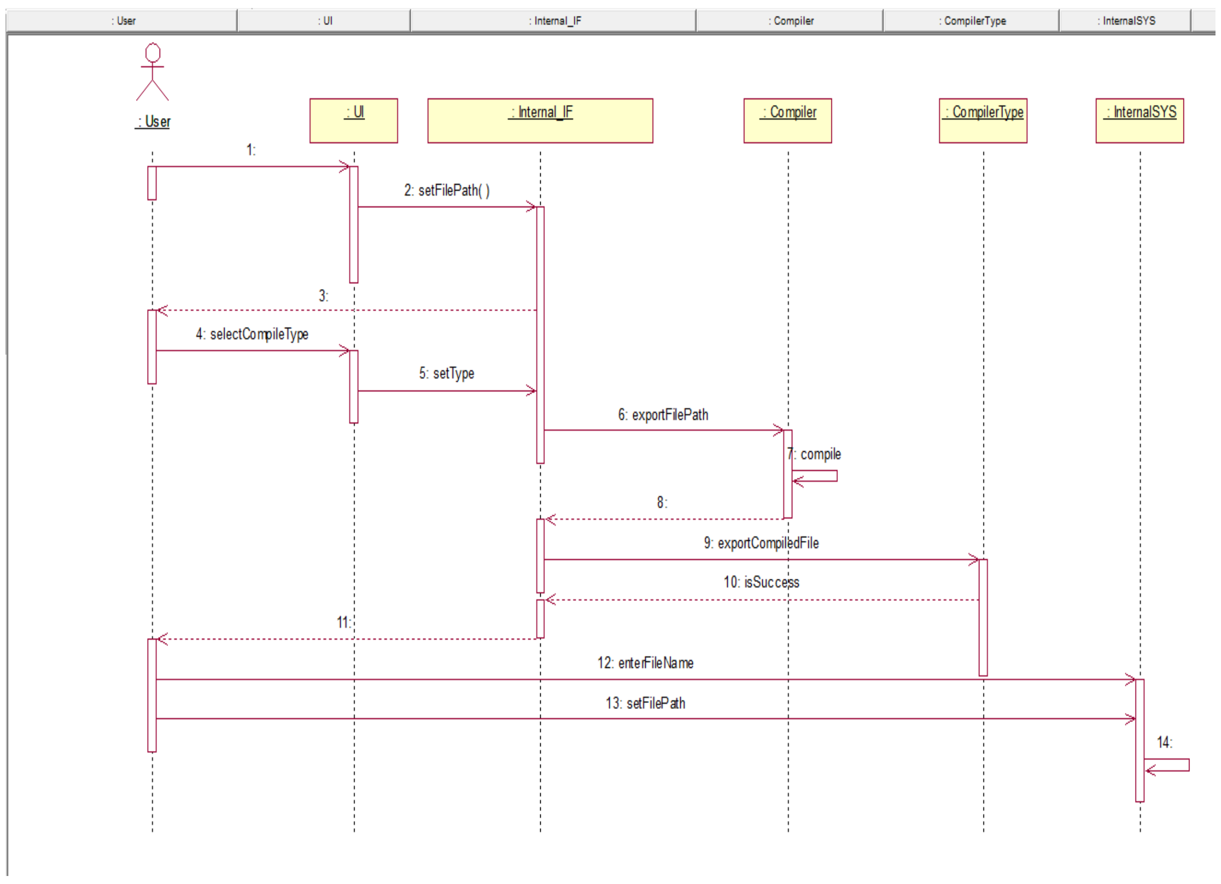
Import file



Select Type



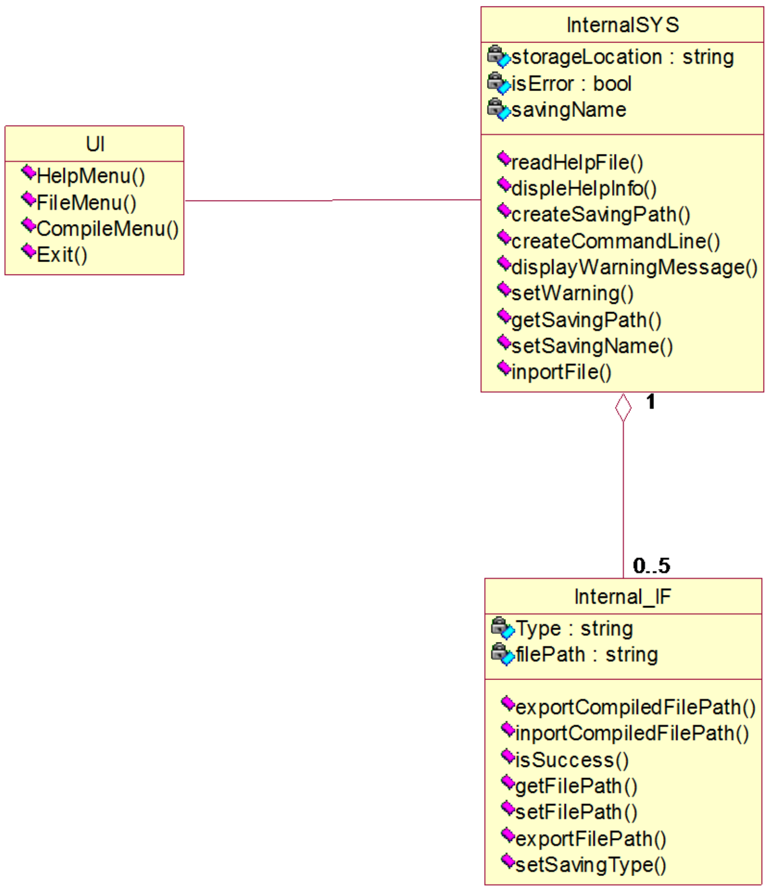
SaveFile



: User
: UI
: Internal_IF
: Compiler
: CompilerType
: InternalSYS
1:
2: setFilePath( )
3:
4: selectCompileType
5: setType
6: exportFilePath
7: compile
8:
9: exportCompiledFile
10: isSuccess
11:
12: enterFileName
13: setFilePath
14:

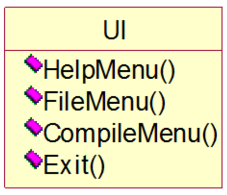
Class Diagram

Internal System



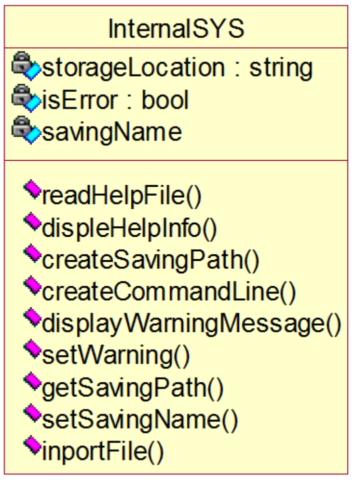
Class Description

Class: UI



Class: UI	
Attribute&Operation	**Description**
HelpMenu()	Display the help information about how to use this system.
FileMenu()	It is used to display the import file and exit button.
CompileMenu()	This will let user to choose the output file type which includes graph, “sql” file, “xml &dtd” file and “xml schema” file.
Exit():	Close software.

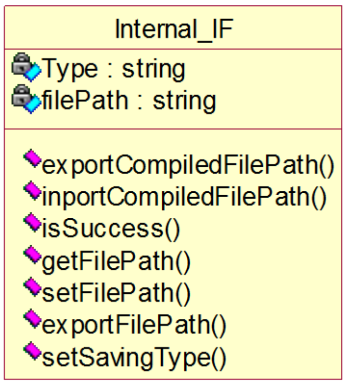
Class: InternalSYS



Class: InternalSYS	
Attribute&Operation	**Description**
<Attribute>	
storagelocation	Save the storage location information.
isError	Save error status.
savingName	Save final file name.
<Operation>	
readHelpFile()	This will read help file.
displayHelpInfo()	This will display help information after read help file.
createSavingPate()	If user wants to change storage location, the system will call this function to create a new storage location.
createCommandLine()	Create command line.
displayWarningMessage()	The system displays error message.
setWarning()	If there have an error, the system will set warning status as true.
getSavingPath()	System gets storage location.
setSavingName()	System set final file’s name.

inportFile()	Get file name and path.

Class: Intenal_IF

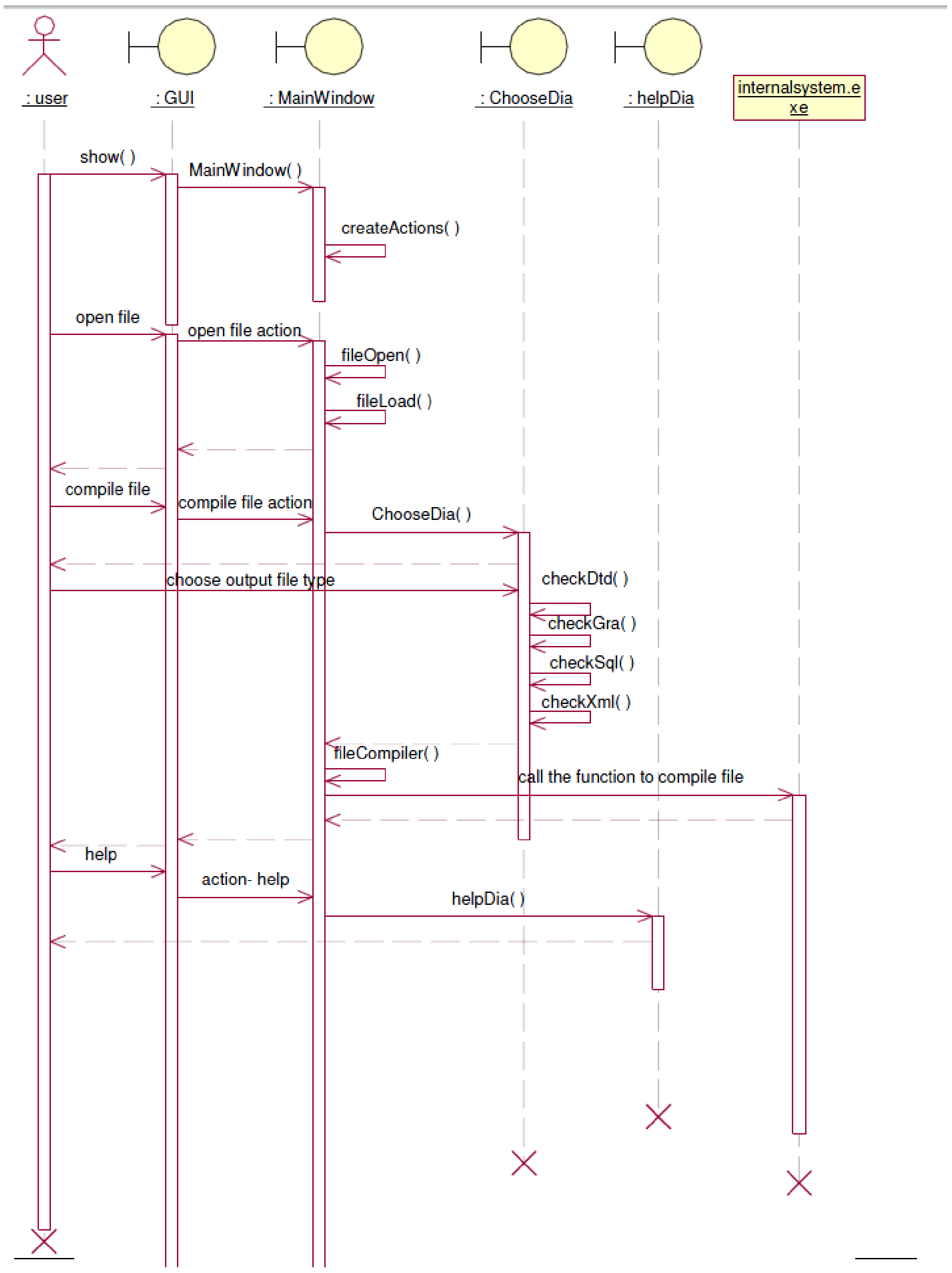


Class: Internal_IF	
Attribute&Operation	**Description**
<Attribute>	
Type	Save output file type which includes graph, “sql” file, “xml&dtd” file and “xml schema” file.
filePath	Save use’s file path.
<Operation>	
exportCompiledFilePath():	This is the interface to let other program to use this file information.
inportCompiledFilePath():	The interface to get the file information from compiler.
isSuccess():	If there is no error, this will return true. else, this will return false.
getFilePath():	Get user’s file path.
setFilePath():	Save user’s file path.
exportFilePath():	This is the interface to send user’s file path and name.
setSavingType():	This will set output file type which includes graph, “sql” file, “xml &dtd” file and “xml schema” file.

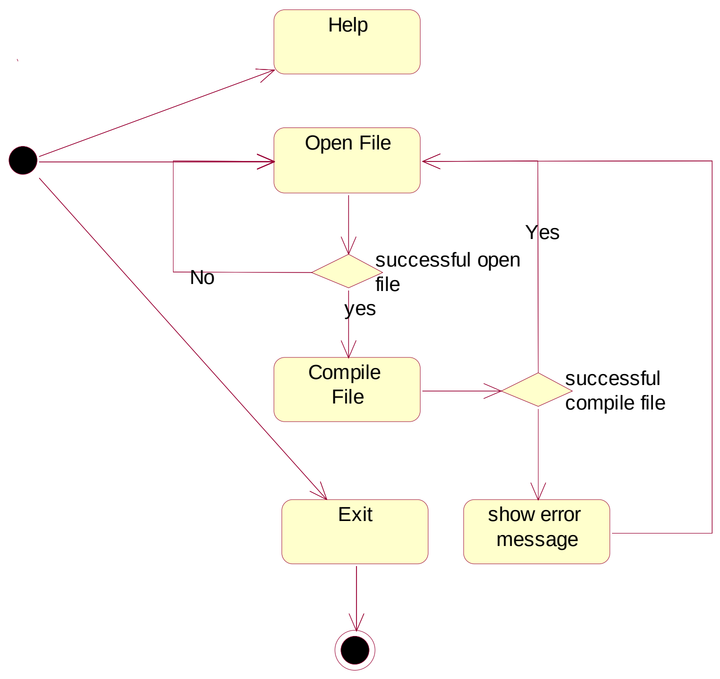
GUI Package

Sequence Diagram

GUI

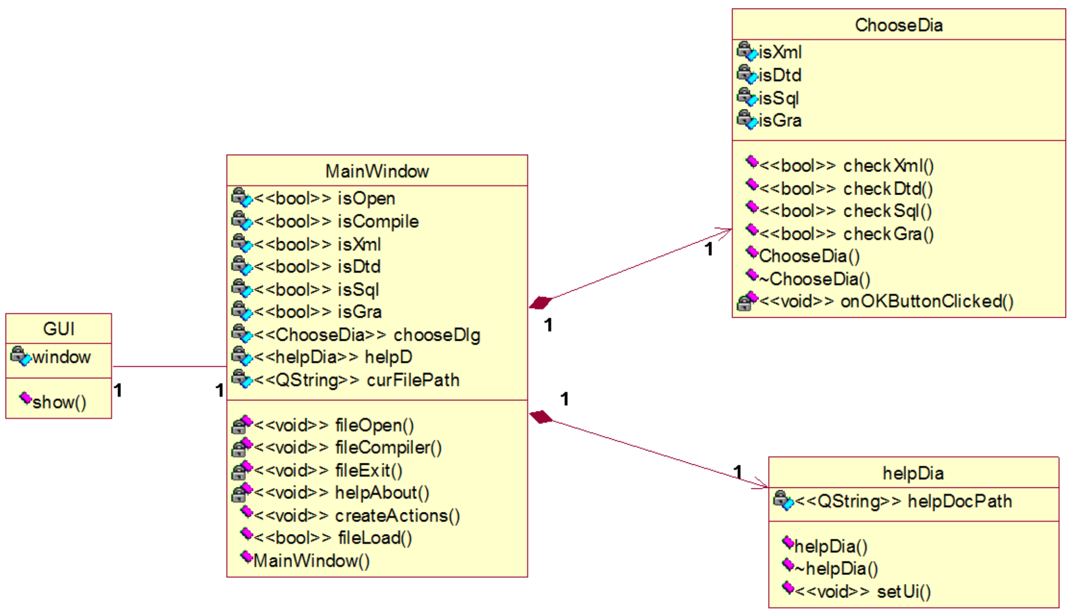


Statement Diagram



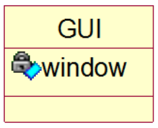
Class Diagram

GUI(Graph User Interface)



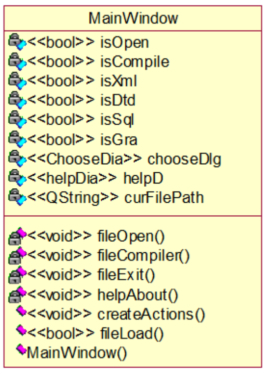
Class Description

Class: GUI



Class: GUI	
Attribute	**Description**
window	Create the Main Window for the user, and show the window to the user

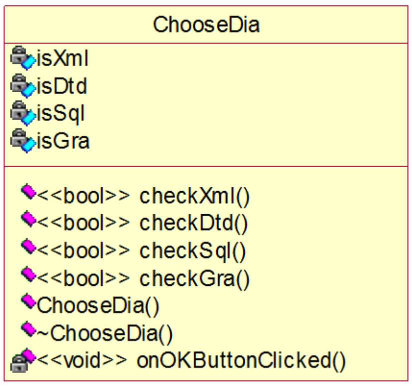
Class: MainWindow



Class: MainWindow	
Attribute&Operation	**Description**
<Attribute>	
isOpen	save the state of the system whether open the file or not
isCompile	Save the state of the system whether compile the file or not
isXml	Save the user choose whether need the XML schema file output or not
isDtd	Save the user choose whether need the DTD file output or not
isSql	Save the user choose whether need the Sql file output or not
isGra	Save the user choose whether need the graph file output or not
<Operation>	
fileOpen()	Get the file path and call the function fileLoad() to open the file, show on the main window area.
fileCompiler()	Show the choose dialog to the user, get the user's choose. Sent the file's path and the user's choose to the "internalsystem.exe" to compile the file. Get the return information and show to the user

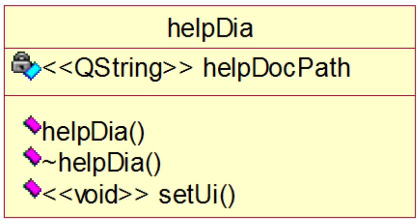
	whether successful compile the file.
fileExit()	Exit the system.
helpAbout()	Show the help dialog to the user.
createAction()	Set the main window’s button’s action.
fileLoad()	Open the file and show the file detail on the main window area, and set the main window title being the path of the file.
MainWindow()	Set up the graph window

Class: ChooseData



Class: ChooseDia	
Attribute&Operation	**Description**
<Attribute>	
isXml	Record the user choose the xml schema file output or not
isDtd	Record the user choose the DTD file output or not
isSql	Record the user choose the SQL file output or not
isGra	Record the user choose the graph file output or not
<Operation>	
checkXml()	Check and return whether the user choose the xml schema file output
checkSql()	Check and return whether the user choose the SQL file output
checkDtd()	Check and return whether the user choose the DTD file output
checkGra()	Check and return whether the user choose the graph file output
ChooseDia()	Set up the choose dialog
~ChooseDia()	Delete the choose dialog
onOkButtonClicked()	Accept the user’s choose, and return to the main window

Class: Help



Class:	
Attribute&Operation	**Description**
<Attribute>	
helpDocPath	Record the help.txt file path
<Operation>	
helpDia()	Call setUi() set up the dialog and load the help.txt to the dialog
setUi()	Initial the help dialog graph
~helpDia()	Delete the help dialog

Core Compile System Design

System Description

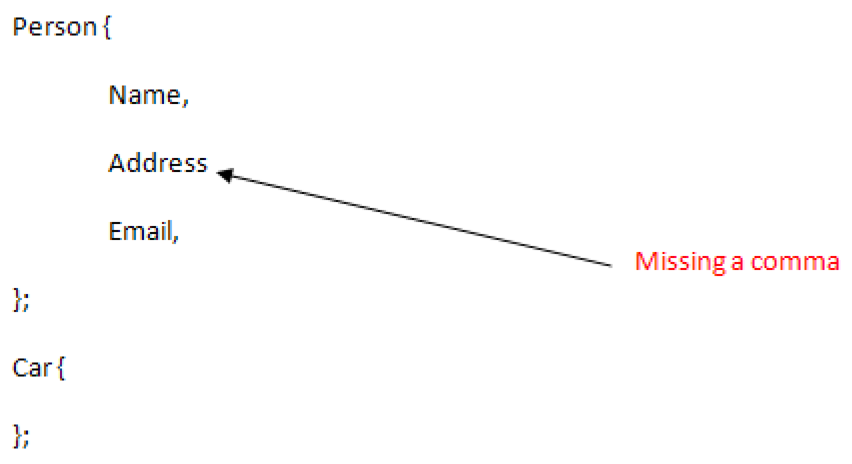
Introduction

The compiler of advanced database design system is designed for analysing syntax and semantic structure of source code from the user, which the syntax rules are based on the Textual Conceptual Modelling Language. It will generate an xml file and store the information inside this file when the syntax from the input file is correct. It is designed as an adjunct system which is used for cooperating with other components of advanced database design system. It also can be added to those systems which want to analyse the syntax of this language and output the information in xml format in the future.

The compiler eliminates and reduces the time of user to detect the syntax errors. If any errors have been detected, compiler will generate a file called errorMessage.txt.

Operational Scenarios

The full path name of the source code file should be given when the compiler is executed. After the execution, it will read the content from the source file, run source code and generate the output file in xml format when the source code has been run successfully. Otherwise, the compiler will return error messages that describe the details of the error to the caller.



```
Person {
        Name,
        Address
        Email,
};
Car {
};
```

In this case, an error message that indicating line number and error type should be created,

Then the compiling process will skip to the next class.

System Requirements

1. The caller must pass the full path name of the source file to the compiler when it is called.
2. The source code should be written in unnamed textual conceptual modelling language.

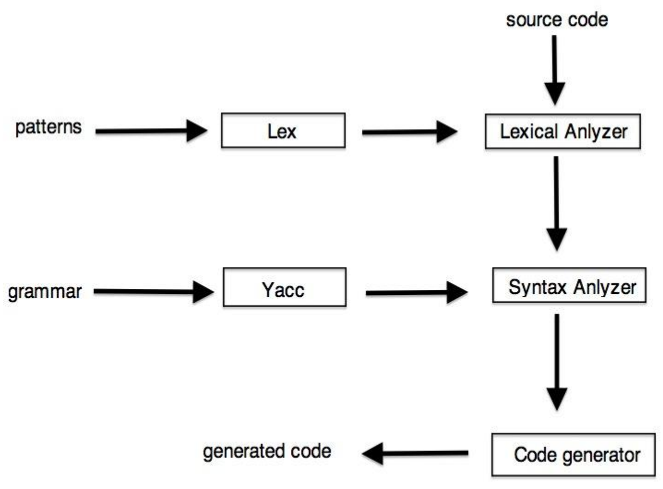
Conceptual Design

The compiler is developed by using Lex&Yacc. Lex is used to implement lexical analyser which can break up the source code into usable tokens excluding comments and whitespace, then store them into a symbol table. Yacc is used for building syntax analyser which can detect both syntax and grammar errors.

In programs with structured input, the source code will be operated with two steps. First, the source code will be divided into meaningful units, and secondly, and then the relationship among the units will be analysed.

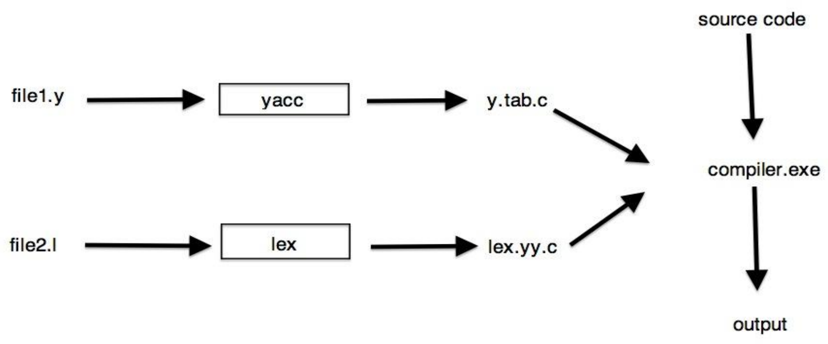
In the first step, known as lexical analyzer, will help us to identify the meaningful token in the future by taking a set of descriptions of possible token (pattern) producing C routine code.

As the input is divided into tokens, our compiler will establish the relationship among the tokens, and we use syntax analyzer to parse the token by searching the expressions, statements and declarations. And syntax analyzer is implemented Yacc through the grammar has been previously defined. Syntax analyzer will detect the input tokens match which rule in the grammar automatically and it also can detect the syntax error which the input token does not match any rules.



B.4-1 lex and yacc

when the lex and yacc code have been successfully run, it will generate a .c file called lex.yy.c and y.tab.c. And we compile these two file and make the compiler executable.



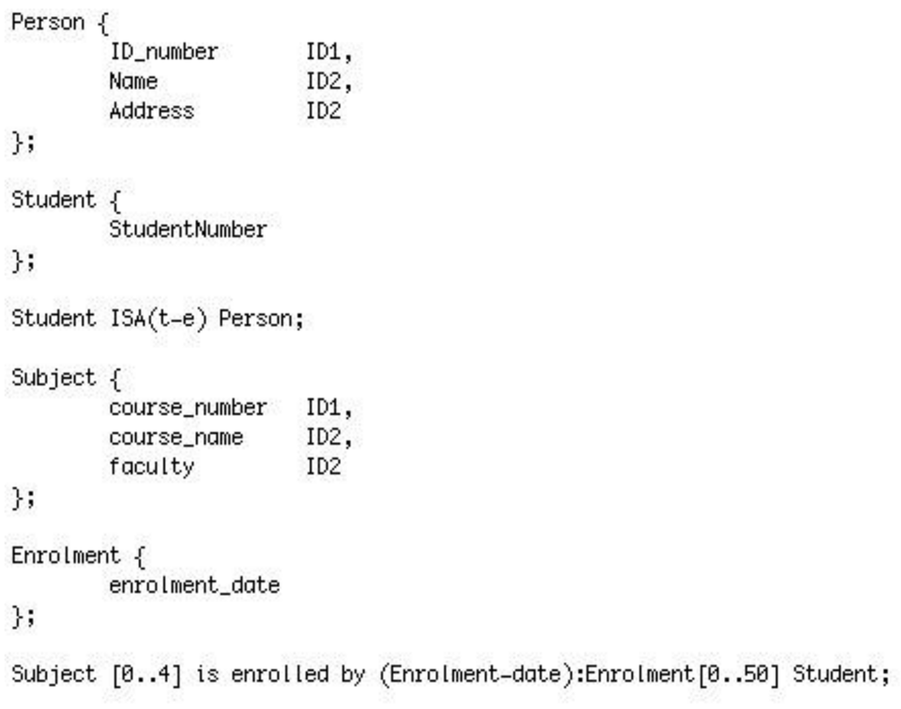
B.4-2 lex and yacc

Input formats should match the grammar of the unnamed textual conceptual modelling language.

Format of input and output document

The output format will classify into three types, type class is recorded the definition of the class, such as attributes, id. Type association is the relationship between classes, in order to make it clear for the further implementation, there are maximum classes involve in one association class object. And generalization is for describe the generalization between two classes.

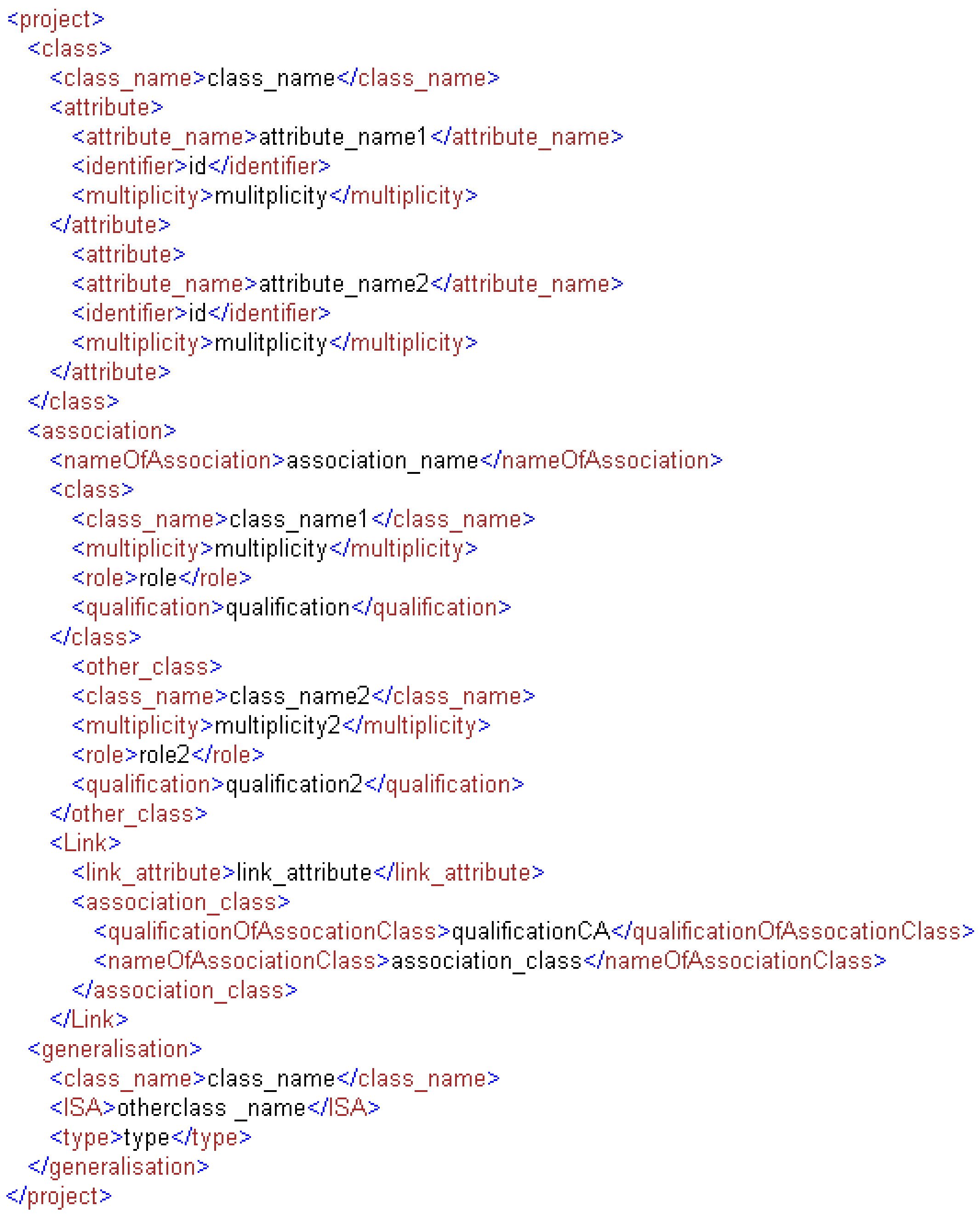
Input format example:



```
Person {
        ID_number       ID1,
        Name            ID2,
        Address         ID2
};

Student {
        StudentNumber
};

Student ISA(t-e) Person;

Subject {
        course_number   ID1,
        course_name     ID2,
        faculty         ID2
};

Enrolment {
        enrolment_date
};

Subject [0..4] is enrolled by (Enrolment-date):Enrolment[0..50] Student;
```

B.4.3 input format example

Output format:



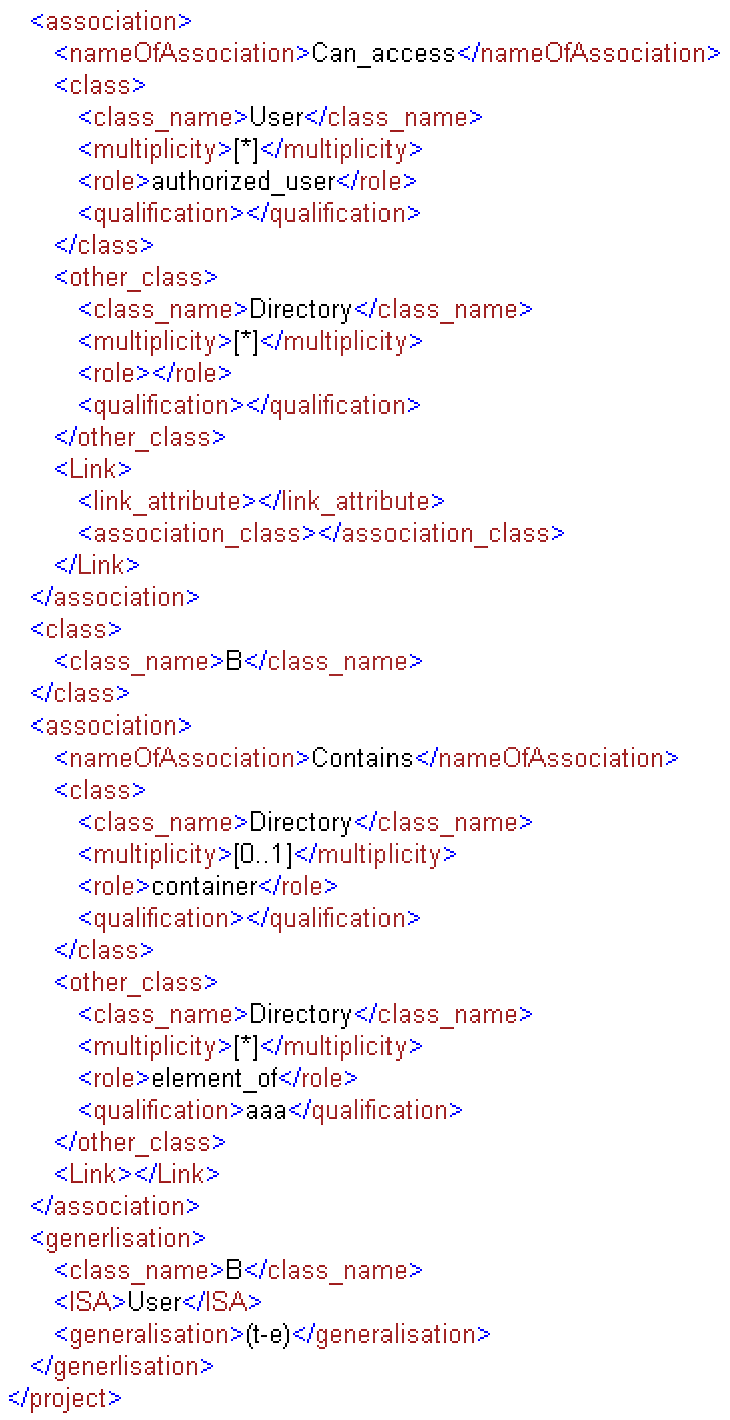
```
<project>
  <class>
    <class_name>class_name</class_name>
    <attribute>
      <attribute_name>attribute_name1</attribute_name>
      <identifier>id</identifier>
      <multiplicity>mulitplicity</multiplicity>
    </attribute>
      <attribute>
      <attribute_name>attribute_name2</attribute_name>
      <identifier>id</identifier>
      <multiplicity>mulitplicity</multiplicity>
    </attribute>
  </class>
  <association>
    <nameOfAssociation>association_name</nameOfAssociation>
    <class>
      <class_name>class_name1</class_name>
      <multiplicity>multiplicity</multiplicity>
      <role>role</role>
      <qualification>qualification</qualification>
    </class>
      <other_class>
      <class_name>class_name2</class_name>
      <multiplicity>multiplicity2</multiplicity>
      <role>role2</role>
      <qualification>qualification2</qualification>
    </other_class>
    <Link>
      <link_attribute>link_attribute</link_attribute>
      <association_class>
        <qualificationOfAssocationClass>qualificationCA</qualificationOfAssocationClass>
        <nameOfAssociationClass>association_class</nameOfAssociationClass>
      </association_class>
    </Link>
  <generalisation>
    <class_name>class_name</class_name>
    <ISA>otherclass _name</ISA>
    <type>type</type>
  </generalisation>
</project>
```

B.4-3 output format

Output format example:

```xml
<project>
  <class>
    <class_name>User</class_name>
    <attribute>
      <attribute_name>uid</attribute_name>
      <identifier>ID1</identifier>
      <multiplicity>[1..5]</multiplicity>
    </attribute>
    <attribute>
      <attribute_name>password</attribute_name>
      <identifier></identifier>
      <multiplicity></multiplicity>
    </attribute>
  </class>
  <class>
    <class_name>Directory</class_name>
    <attribute>
      <attribute_name>path</attribute_name>
      <identifier>ID</identifier>
      <multiplicity></multiplicity>
    </attribute>
    <attribute>
      <attribute_name>dname</attribute_name>
      <identifier>ID</identifier>
      <multiplicity></multiplicity>
    </attribute>
    <attribute>
      <attribute_name>tot_files</attribute_name>
      <identifier></identifier>
      <multiplicity></multiplicity>
    </attribute>
  </class>
  <association>
    <nameOfAssociation>Owns</nameOfAssociation>
    <class>
      <class_name>User</class_name>
      <multiplicity></multiplicity>
      <role>owner</role>
      <qualification></qualification>
    </class>
    <other_class>
      <class_name>Directory</class_name>
      <multiplicity>[*]</multiplicity>
      <role></role>
      <qualification></qualification>
    </other_class>
    <Link>
      <link_attribute></link_attribute>
      <association_class></association_class>
    </Link>
  </association>
```

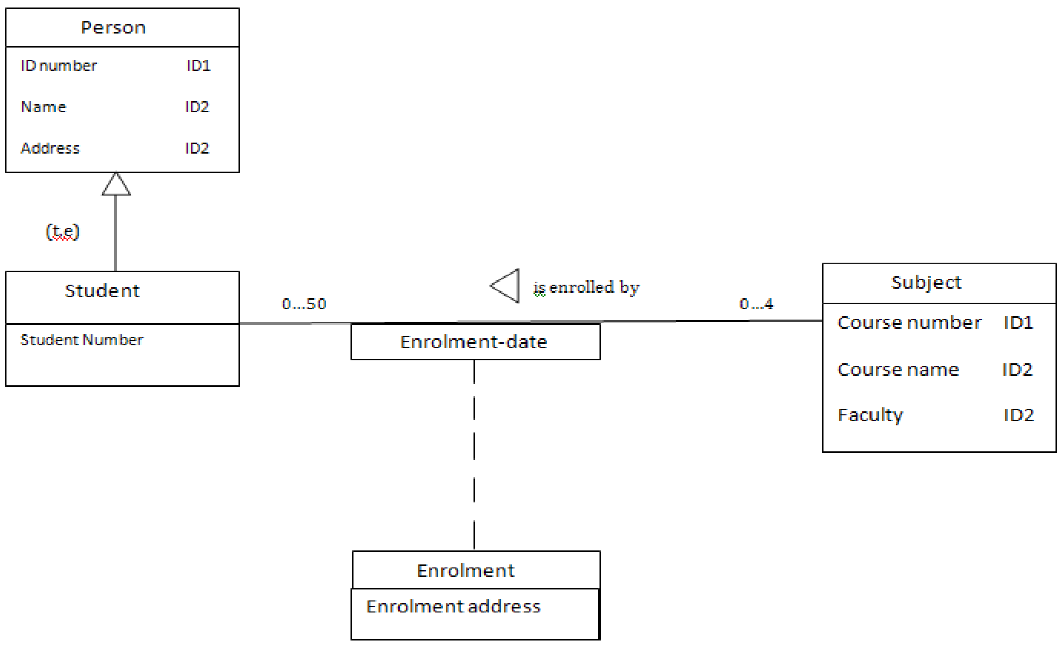
B.4-4 output example



```
  <association>
    <nameOfAssociation>Can_access</nameOfAssociation>
    <class>
      <class_name>User</class_name>
      <multiplicity>[*]</multiplicity>
      <role>authorized_user</role>
      <qualification></qualification>
    </class>
    <other_class>
      <class_name>Directory</class_name>
      <multiplicity>[*]</multiplicity>
      <role></role>
      <qualification></qualification>
    </other_class>
    <Link>
      <link_attribute></link_attribute>
      <association_class></association_class>
    </Link>
  </association>
  <class>
    <class_name>B</class_name>
  </class>
  <association>
    <nameOfAssociation>Contains</nameOfAssociation>
    <class>
      <class_name>Directory</class_name>
      <multiplicity>[0..1]</multiplicity>
      <role>container</role>
      <qualification></qualification>
    </class>
    <other_class>
      <class_name>Directory</class_name>
      <multiplicity>[*]</multiplicity>
      <role>element_of</role>
      <qualification>aaa</qualification>
    </other_class>
    <Link></Link>
  </association>
  <generlisation>
    <class_name>B</class_name>
    <ISA>User</ISA>
    <generalisation>(t-e)</generalisation>
  </generlisation>
</project>
```

B.4-5 output example

B.4-4 and B.4-5 represent the association among the classes as follow,



Response to identifiable error conditions

When the name of file has been passed to compiler is invalid, the compiler will be terminated and sent the failure signal to controller.

I f the syntax error has been detected, compiler will skip remain part of class definition, and analyze the next class definition. And display all the error message and sent a failure message to controller later.

Response to identifiable failure conditions

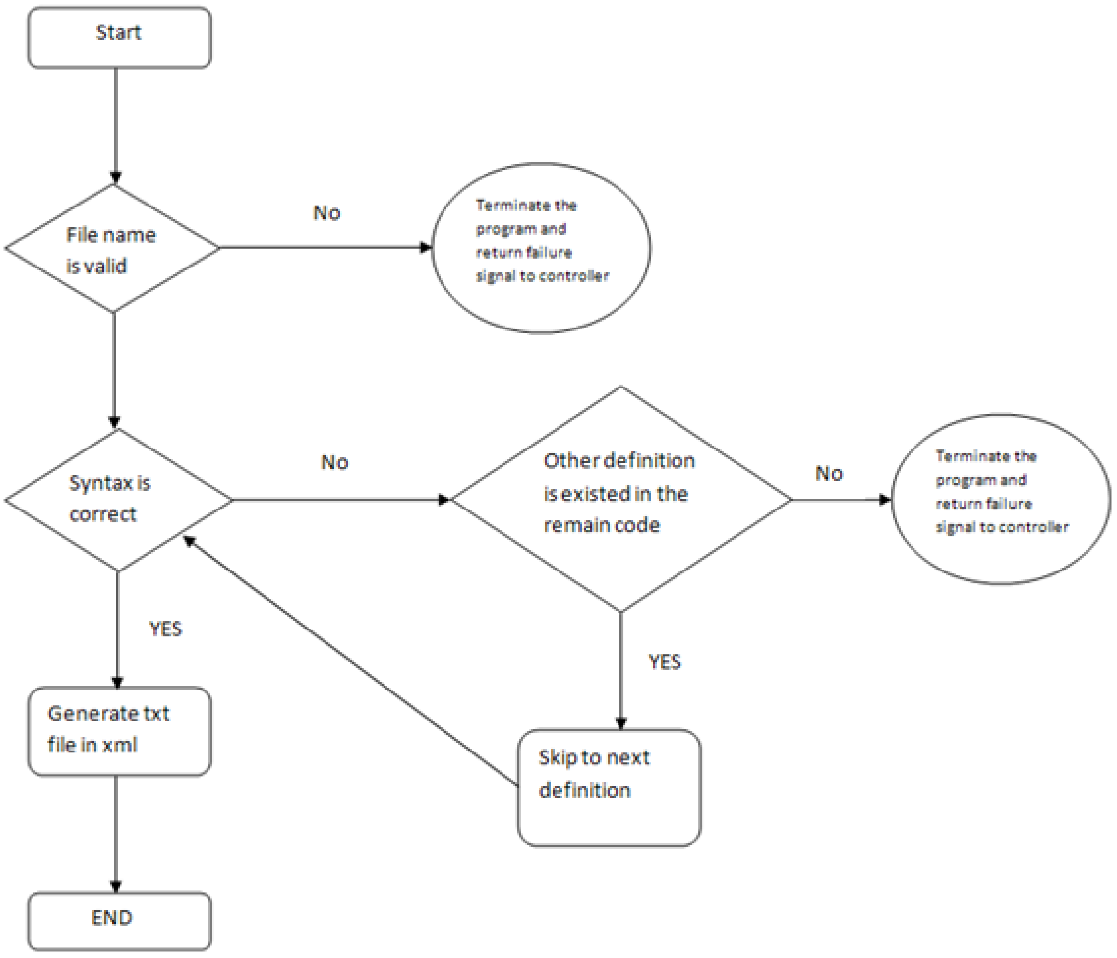
Compiler will display a error message when the name of source file is invalid. And when the logic error has been detected, such as cycle among the class and duplicated associations, compiler will display a warning message

Assumptions Made

1. The output format of compiler might change in the future, in order to fit for the tool that can extract data, and increase the efficiency of other subsystem to read the data from the information text file.

2. Warning message will be display when the circle is generated from the user's source code, and more warning message will be concerned in the future implementation.

Data flow diagram


Start
File name is valid
No
Terminate the program and return failure signal to controller
Syntax is correct
No
Other definition is existed in the remain code
No
Terminate the program and return failure signal to controller
YES
YES
Generate txt file in xml
Skip to next definition
END

Software Design

The software requirements and overview have been dealt with elsewhere in this document. The present section addresses the design and implementation of the software that forms the advanced database design system.

Software Development Environment

- Language of implementation:
  name : Lex and Yet Another Compiler -Compiler(Yacc)

  description : language for implement compiler

- Tool for implementation:
  name : Flex

  version : 2.5.4a

  description: tool for compile lex source file

  name : Bison

  version : 2.4.1

  description : tool for compile yacc source file

Software Quality Assurance

We will test our software by using black and white box test, and we also will do the unit testing and system testing, in order the assure the compiler can run interdependently and collaborate with other component inside the system, all test cases will use for each version of the compiler to avoid the regression fault. During the design, we will used agile method, we will deliver the source code to our supervisor every 2 weeks to make sure the quality of the project.

Preconditions for Software

Preconditions for System Startup

None

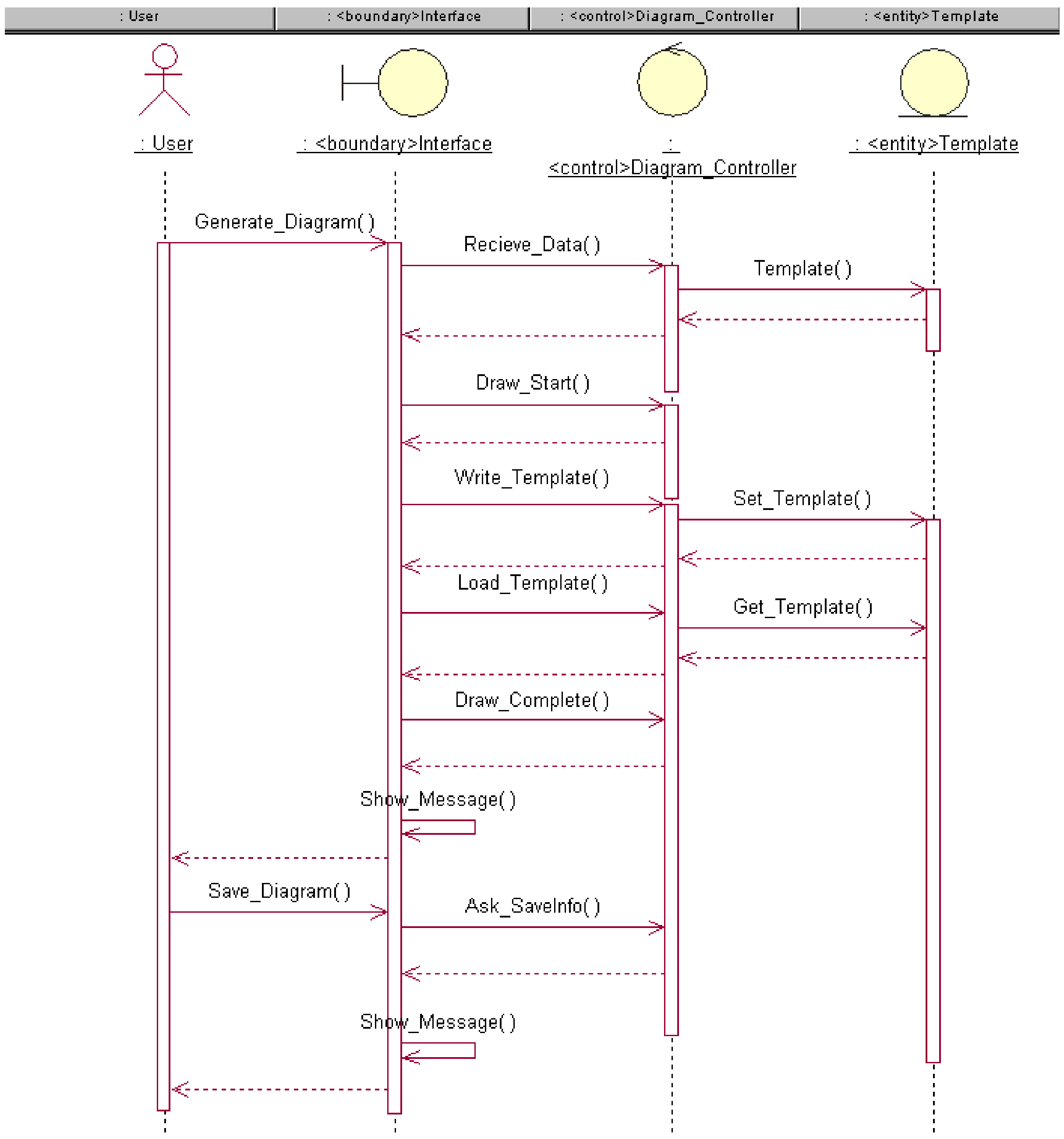
Preconditions for System Shutdown

None

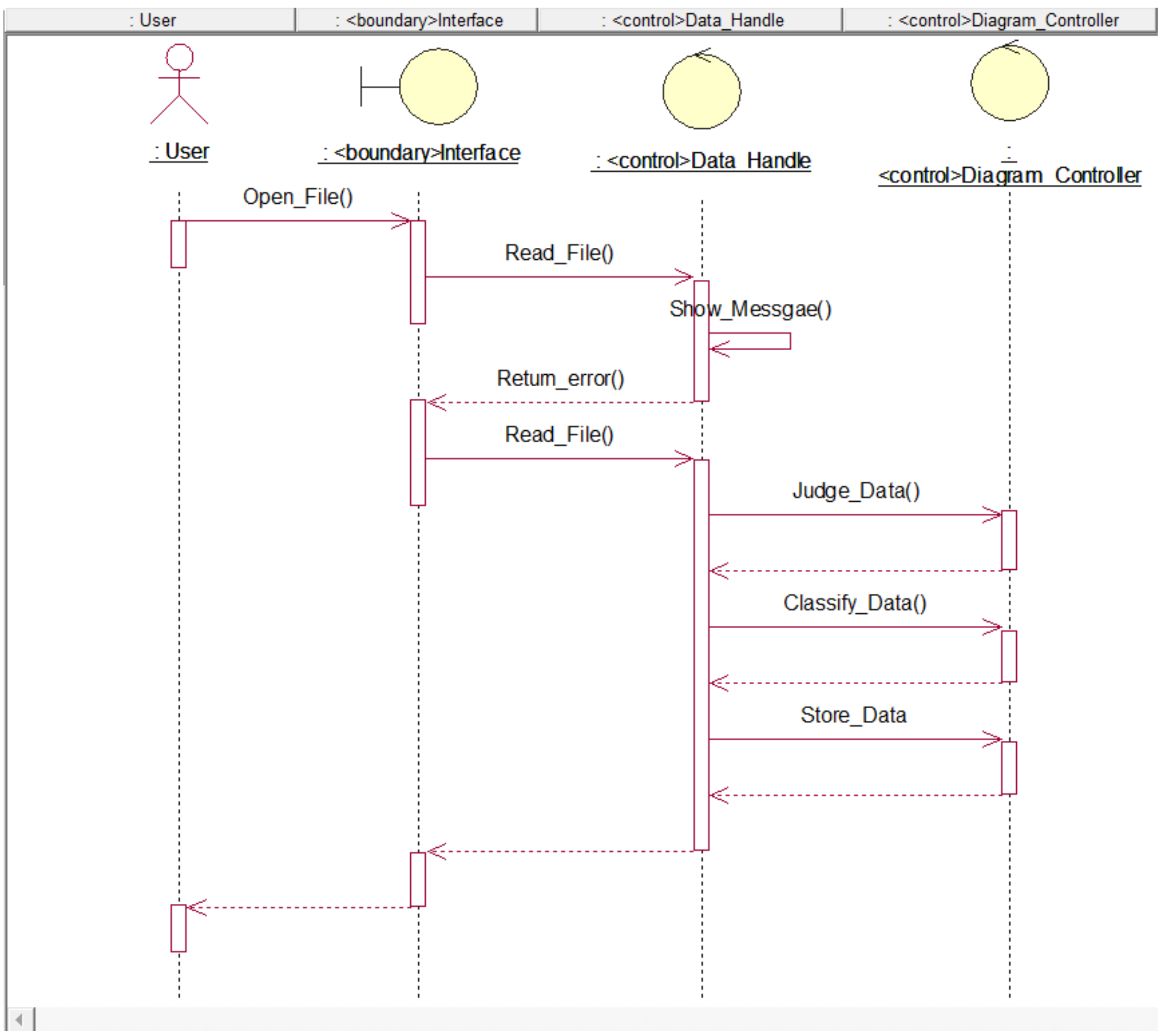
UML Class Diagram Generate

Sequence Digram

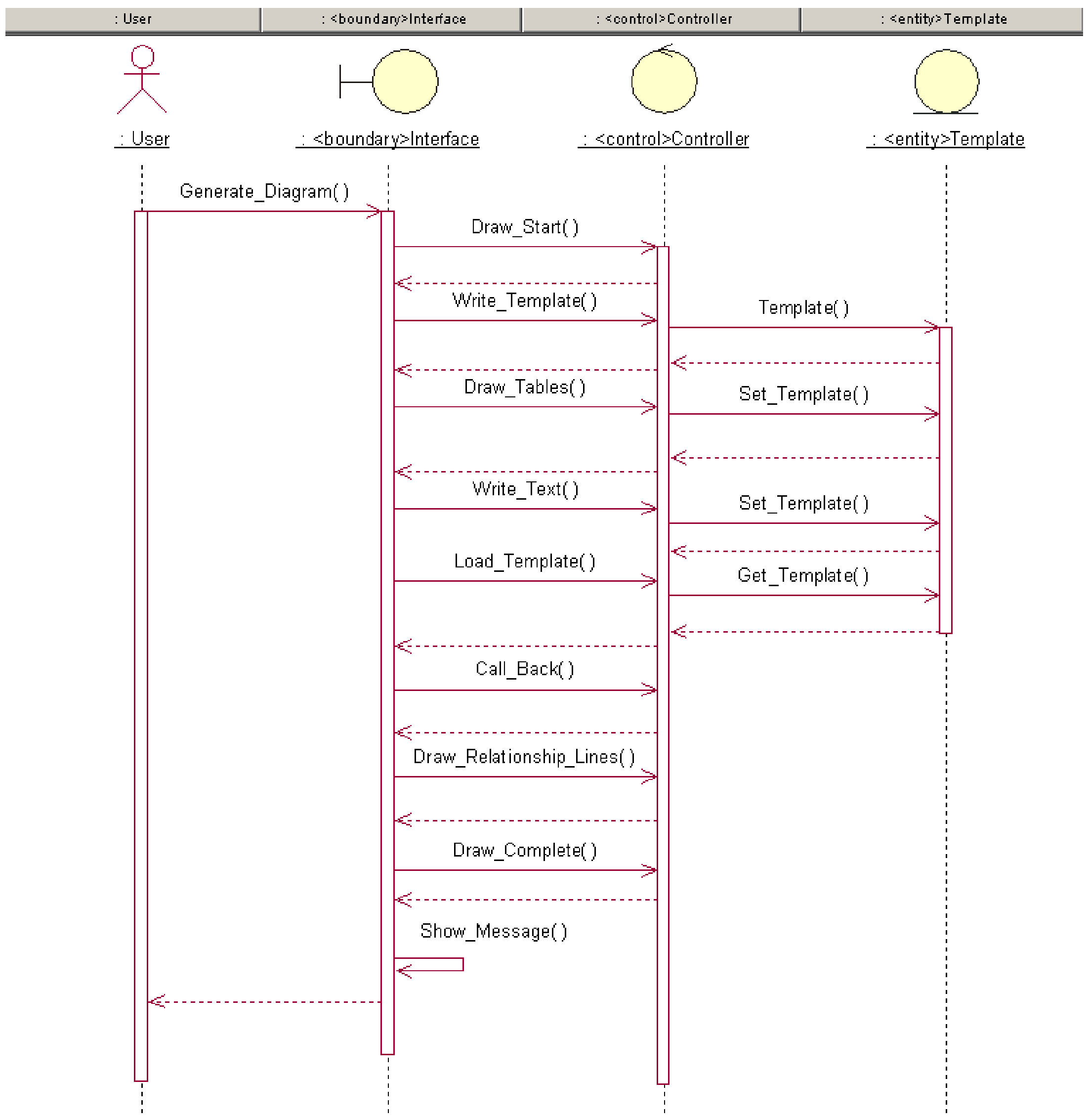
Diagram Generate



Data Dispose

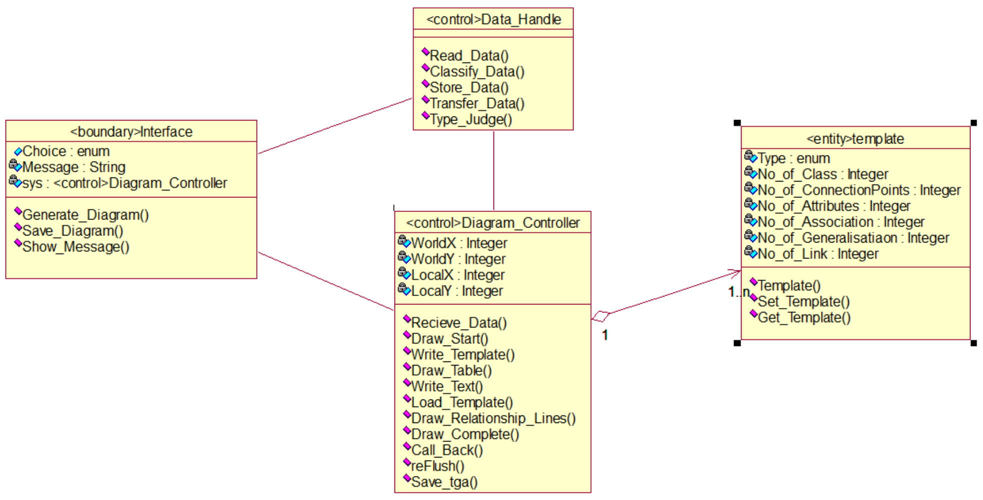


Template Working



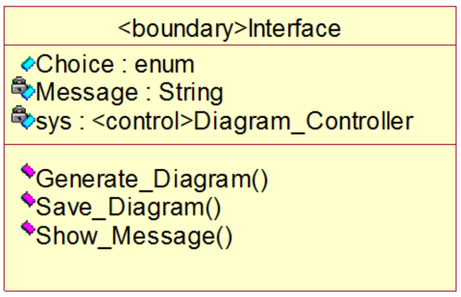
Class Diagram

Diagram Generate



Class Description

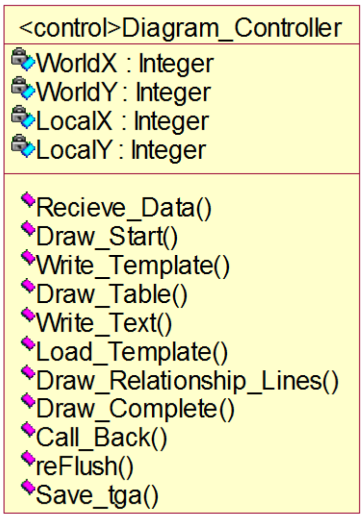
Class: Interface



Class: Interface	
Attribute&Operation	**Description**
<Attribute>	
Choice	User could choose the scheme type which user want(include UML class diagram, SQL table and XML&DTD), the value type is enum.

Messag	The software could return the message weather the scheme generated successful or not
<Operation>	
Generate_Diagram()	After the file compiled completely, user can generate the result scheme user want include UML class diagram, SQL table and XML&DTD scheme. The type of choice is enum.
Show_Message()	After generate, the system display the message to use weather the file generated successful or not.
Save_Diagram()	User could save the UML diagram as "bmp" form. And the default system generate form is "bmp".

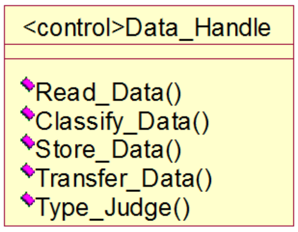
Class: Diagram Generate



Class: Diagram_Controller	
Attribute&Operation	**Description**
<Attribute>	
WorldX	The basic world x-coordinate of screen
WorldY	The basic world y-coordinate of screen
LocalX	The actually x-coordinate of object (class box, lines, text and etc)
LocalY	The actually y-coordinate of object (class box, lines, text and etc)
<Operation>	
Receive_Data()	When user click the "Generate UML Class Diagram" button, the receive function should get data from internal system. Then system must read and deal with the data(xml form) which can be implement by OpenGL.

Draw_Start()	Ask software begin to draw the diagram and initialize the whole value which program need.
Write_Template()	The parts of the class diagram has drew and saved in the template, after the data of table got, the value of template should setup and declared.
Draw_Tables()	The function using to draw table with the number of attributes, the table length changed by the increase of attributes . Each connect point must recorded.
Write_Text()	After the class table drew, we should write the table name, attribute name and identifier. The class name layout at centre and aligns text to the left default.
Load_Template()	The DG(Diagram Generate) load the complete template and layout all the template with available space and set in available layout.
Draw_Relationship_Lines()	After two at most three class table loaded, we draw the relationship lines between these tables. Moreover, the lines must not intersect.
Call_Back()	Recall the template function.
Draw_Complete()	Finish the whole draw process and delete all the valuables. Terminate the draw function.
reFlush()	Re-flush the current windows after draw an object
Saving_tga()	Allow user to save the diagram as .tga form, the default name of file is graphical.tga and the save path is below the software directory.

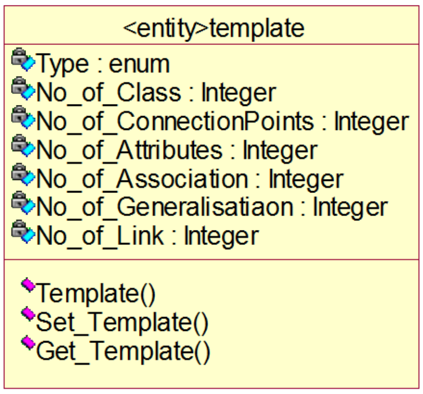
Class: Data_Handle



Class: Template	
Attribute&Operation	**Description**
Read_Data()	Read data from the temp xml file
Classify_Data():	Classify the type of data read in which is class, attribute, association and etc.
Store_Data():	Store data into map depends on different type of data

Transfer_Data()	Transfer different type to the diagram drawing part which include the information of drawing and handle information.
Type_Judge()	Judge the type

Class: Template

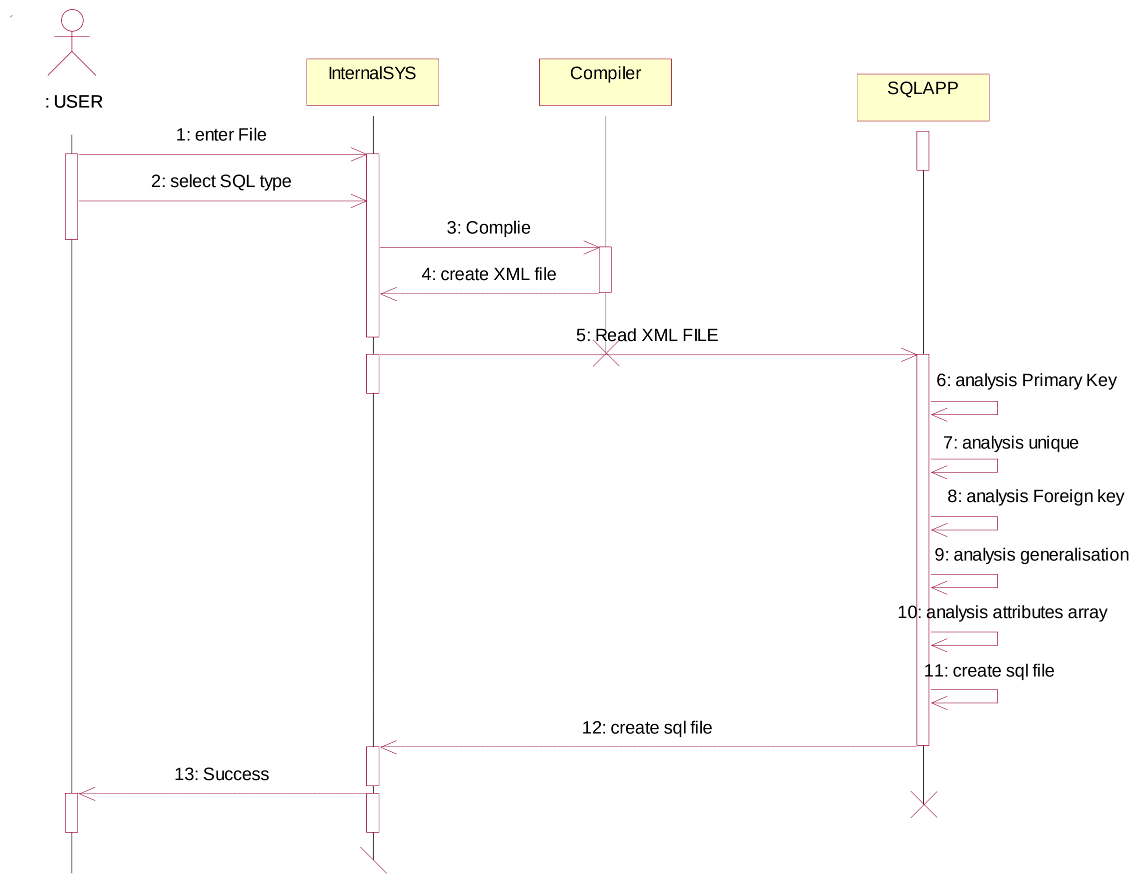

<entity>template
Type : enum
No_of_Class : Integer
No_of_ConnectionPoints : Integer
No_of_Attributes : Integer
No_of_Association : Integer
No_of_Generalisatiaon : Integer
No_of_Link : Integer
Template()
Set_Template()
Get_Template()

Class: Template	
Attribute&Operation	**Description**
<Attribute>	
Type	The type of data which is class, attribute, association and etc.
No_of_Class	The total number of class
No_of_ConnectPoints	The total number of point which connected with other obejct
No_of_Attributes	The total number of attributes in a class
No_of_Association	The total number of association of classes.
No_of_Generalisation	The total number of generalization of classes
No_of_Link	The total number of link of association
<Operation>	
Template().	The constructor of generate and initialize template
Set_Template():	Set all value into the template function include the number of class, number of attributes and number of operations.
Get_Template():	Using load template from template library.

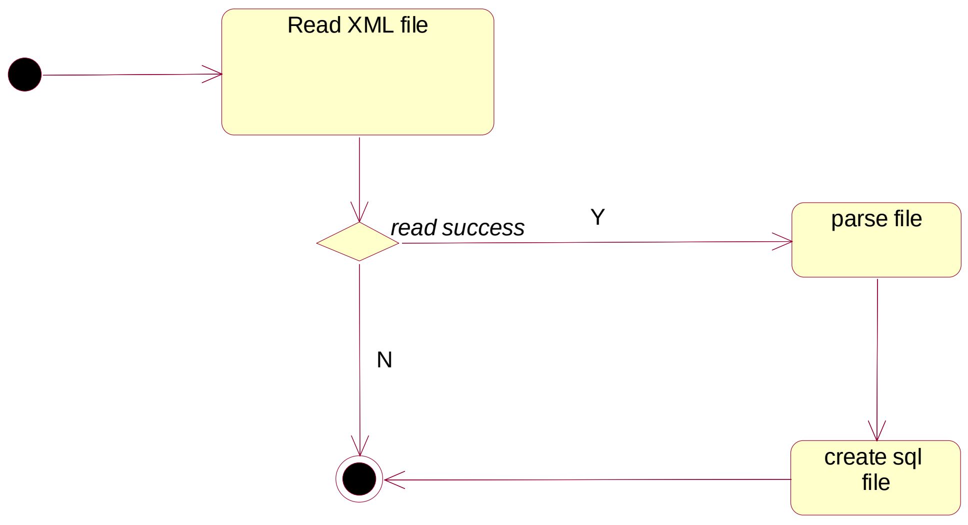
SQL Table Generate

Sequence Diagram

SQL Generate

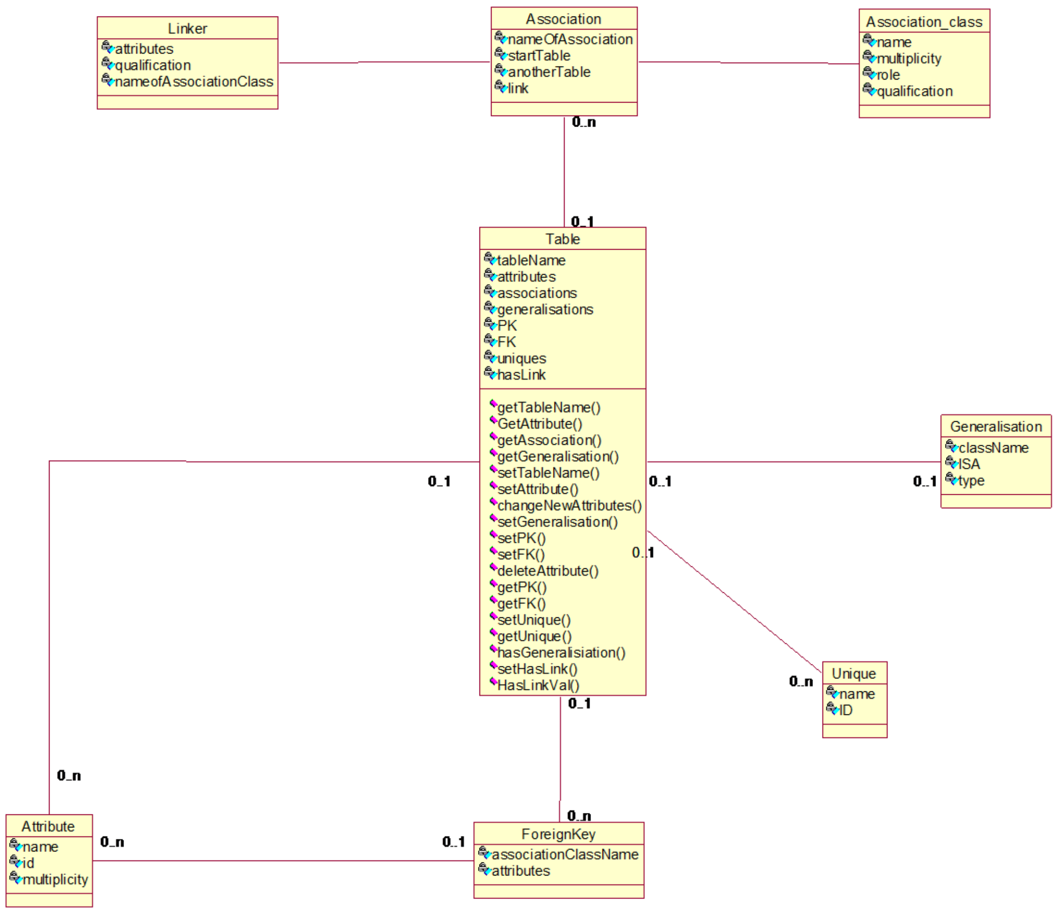


Statement Diagram



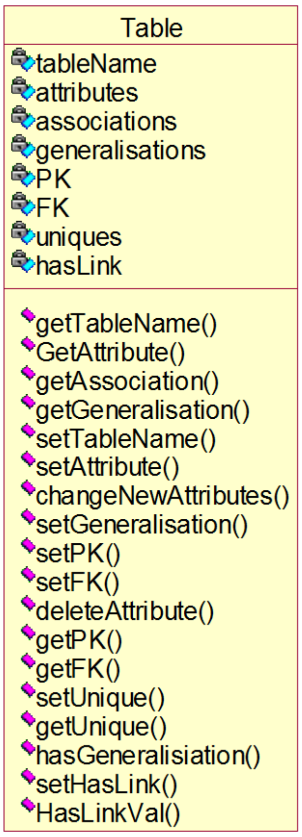
Class Diagram

SQL Generate



Class Description

Class: Table



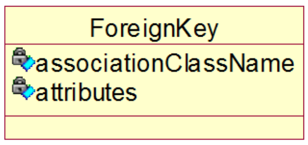
Class: GUI	
attribute	**Description**
tableName	The name of the table
attributes	Attributes in the table
associations	Some association with this table
generalisations	The generalization relationship
PK	Primary key
FK	Foreign key
uniques	The unique of this table
hasLink	This is bool type, decide whether the table has link between another table.
<Operation>	

stringgetTableName();	Get table name, return string type
vector<Attribute>getAttribute();	Get all attributes from table. Return vector<Attribute> type
vector<Association>getAssociation();	
GeneralisationgetGeneralisation();	Get the generalization from table. Return generalization type
voidsetTableName(std::string);	Set table name, has parameter:string

voidsetAttribute(Attribute);
voidsetAttribute(std::vector<Attribute>);
	Set attributes of the table. Has two kinds of parameter:
	1. Attribute
	2. vector<Attribute>

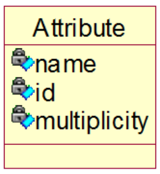
voidsetAssociation(Association);	Set association of the table. has parameter: Association
voidsetGeneralisation(Generalisation);	Set generalization of the table. Has parameter: Generalisation.
voidsetPK();	Set primary key.
voidsetFK(ForeignKey);	Set foreign key, has parameter: ForeignKey
voiddeleteAttribute(std::string);	Delete 1attribute from table. Has parameter: string.
vector<Attribute>getPK();	Get the primary key
vector<ForeignKey>getFK();	Get the foreign key
voidsetUnique();	Set unique
vector<Unique>getUnique();	Get unique
boolhasGeneralisiation();	Decide the generalization type exist in the table
voidsetHasLink(boolval)	Decide the linker exist in the table
boolHasLinkVal()	Get the linker statue. Exist or not

Class: ForeignKey



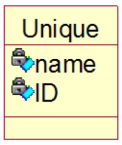
Class: ForeignKey	
Attribute&Operation	**Description**
<Attribute>	
associationClassName	Association class name
attributes	The attributes

Class: Attrubute



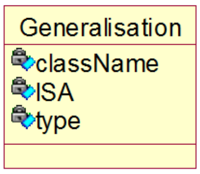
Class: Attribute	
Attribute&Operation	**Description**
<Attribute>	
name	Attribute name
ID	The ID, it used to be decide whether it should be as a primary key.
multiplicity	Numbers of the Attributes

Class: Unique



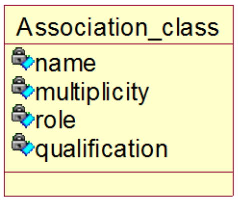
Class: Unique	
Attribute&Operation	**Description**
<Attribute>	
name	Unique name
ID	Unique ID

Class: Generalisation



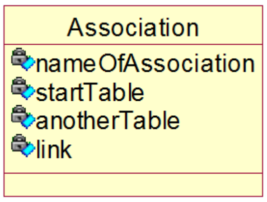
Class: Generalisation	
Attribute&Operation	**Description**
<Attribute>	
calssName	Sub class name
ISA	Super class name
Type	Generalizationtype

Class: Association_class



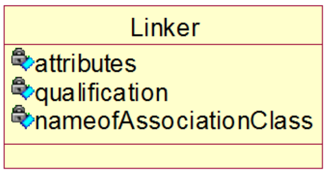
Class: Association_class	
Attribute&Operation	**Description**
<Attribute>	
name	Association class name
multiplicity	multiplicity
role	Relational type
qualification	qualification

Class: Association



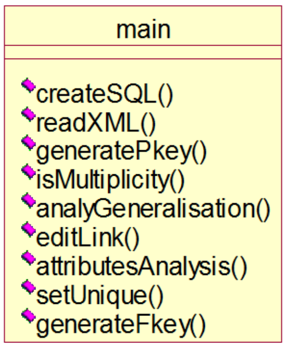
Class: Association	
Attribute&Operation	**Description**
<Attribute>	
nameOfAssociation	Association name
startTable	One side table
anotherTable	Another side table
link	The linker between these two table

Class: Linker



Class: Unique	
Attribute&Operation	**Description**
<Attribute>	
attributes	Linker attributes.
qualification	qualification
nameOfAssociationClass	Association class name

Main.cpp



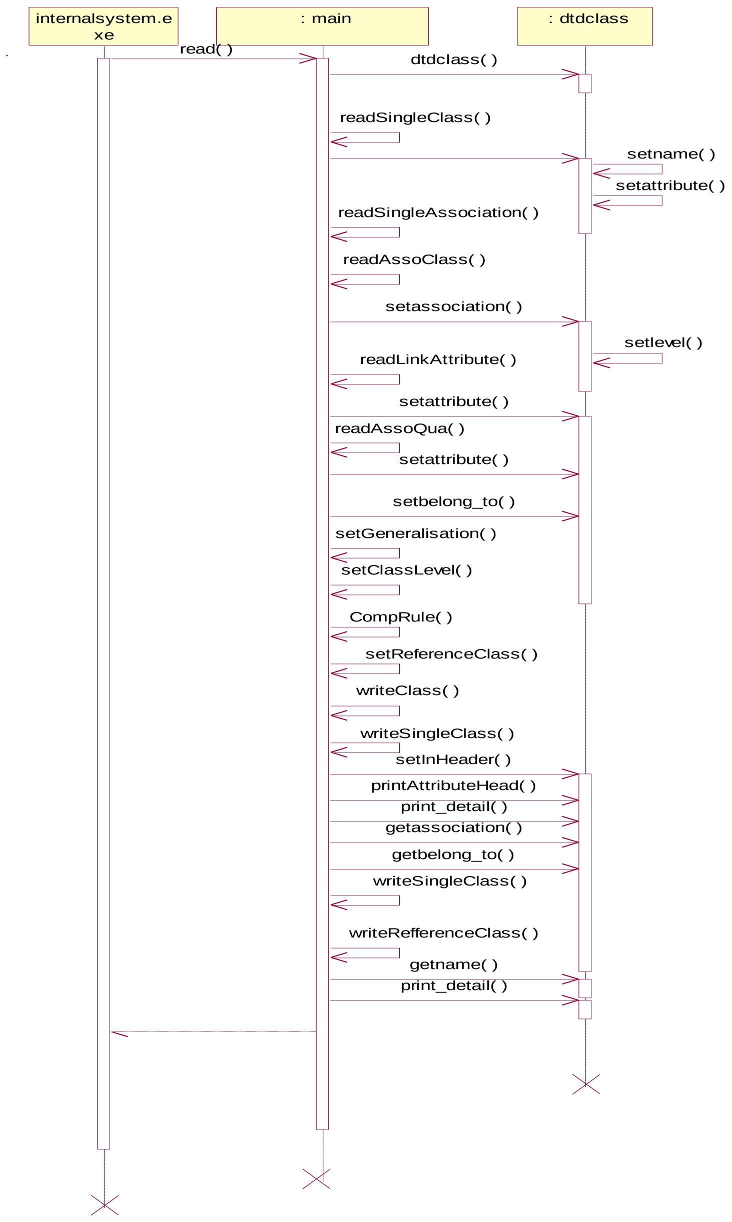
Class:	
Operation	**Description**
<Operation>	
voidcreateSQL(map<string, Table>&)	Create sql file
boolreadXML(map<string, Table>&)	Reading XML file and save data into map container
VoidgenerateFKey(map<string, Table>&)	Analysis foreign key
voidgeneratePKey(map<string,	Analysis primary key

Table>&)	
boolisMultiplicity(Association_class&)	Decide the association class has multiplicity relationship.
voidanalyGeneralisation(map<string, Table>&)	Analysis generalization
voideditLink(map<string, Table>&, Association&)	Create a linker when two table has many to many relation.
voidattributesAnalysis(map<string, Table>&)	Analysis the attribute array
voidsetUnique(map<string,Table>&)	Set unique relation.

DTD Schema Generate

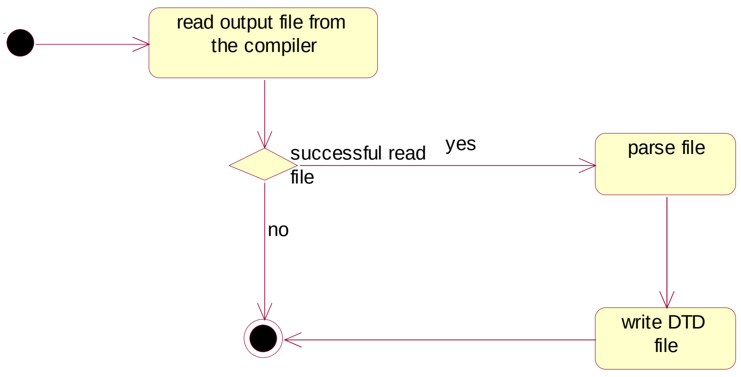
Sequence Diagram

DTD Output



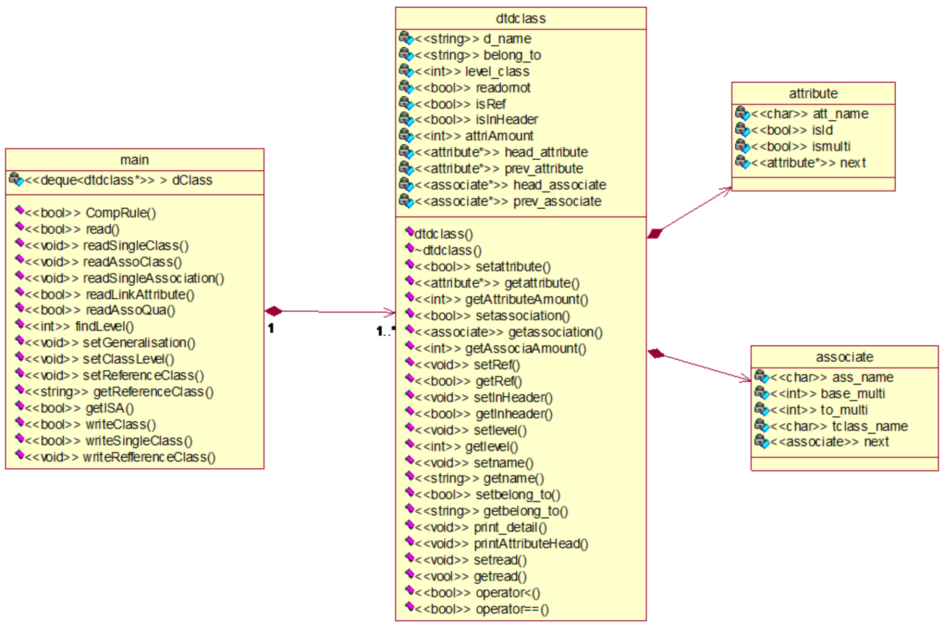
Statement Diagram

DTD Output



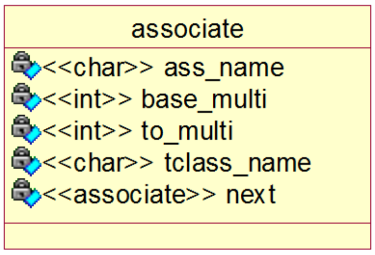
Class Diagram

DTD Generate



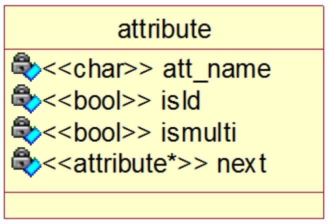
Class Description

Class: associate



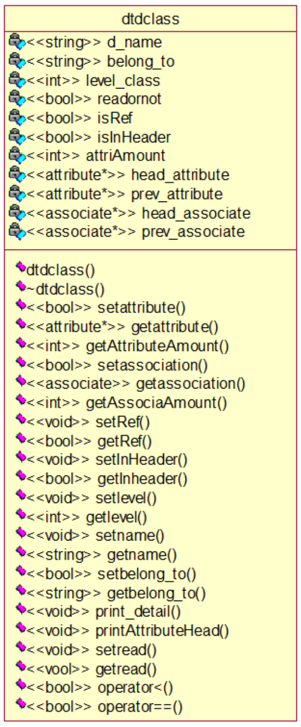
Class:	
Attribute&Operation	**Description**
<Attribute>	
ass_name	Record the name of the association
base_multi	Record the quantity of the class that appear in this association
to_multi	Record the quantity of the relate class that appear in this association
tclass_name	Record the name of the relate class
next	Link to the next association

Class: attribute



Class:	
Attribute&Operation	**Description**
<Attribute>	
att_name	Record the name of the attribute
isId	Record the attribute is ID or not
Ismulti	Record the attribute is multiple or not
next	Link to the next attribute

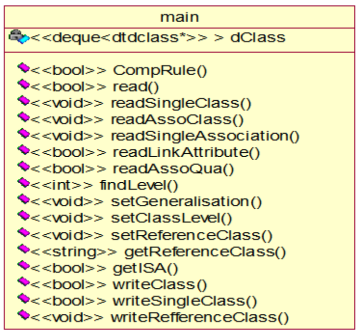
Class: dtdc



Class:	
Attribute&Operation	**Description**
<Attribute>	
d_name	Record the name of the class
belong_to	If the class has generation relationship with other class, record the name of "father" class.
level_class	Record the class level.
readornot	Record whether the class has written into the file already
isRef	Record whether the class is the reference class in the DTD file
isInHeader	Record the class whether has appeared in the other class
attriAmount	Record the number of attribute of the class
associAmount	Record the number of association of the class
head_attribute	Link list of the attribute
prev_attribute	Record the current attribute
head_associate	Link list of the association
prev_associate	Record the current association
<Operation>	
dtdclass()	Initializing the class
~dtdclass()	Delete the class
setattribute()	Set or add the attribute into the class
getattribute()	Each time return the detail of one attribute in the class

getAttributeAmount()	Return the amount of the attributes
setassociation()	Set or add the association into the class
getassociation()	Each time return the detail of one association in the class
getAssociaAmount()	Return the amount of the associations
setRef()	Set the class is the reference class
getRef()	Return the class is the reference class or not
setInHeader()	Set the “isInHeader” attribute being true
getInHeader()	Return the “isInHeader” attribute state
setlevel()	Set the class level for sort inside of the deque, Class occurs 0 time in level 4, occur 1 time in level 1. Class occurs 1 time and the relate class occur more than 2 time set class in level 2. Class occur more than 2 times in level 3
getlevel()	return the level of the class
Setname()	set the name of the class
getname()	return the name of the class
setbelong_to()	set the name of the class of the ISA relationship
getbelong_to()	return the name of the “ISA “ class
print_detail()	Print the declaration of each class attribute into the file. But will not print the attribute which had appear before in the file.
printAttributeHead()	print attribute name in the declaration
setread()	set the class has been write into the DTD file
getread()	return the class has been write into the file or not

Class: Main


main
<<deque<dtdclass*>> > dClass
<<bool>> CompRule()
<<bool>> read()
<<void>> readSingleClass()
<<void>> readAssoClass()
<<void>> readSingleAssociation()
<<bool>> readLinkAttribute()
<<bool>> readAssoQua()
<<int>> findLevel()
<<void>> setGeneralisation()
<<void>> setClassLevel()
<<void>> setReferenceClass()
<<string>> getReferenceClass()
<<bool>> getISA()
<<bool>> writeClass()
<<bool>> writeSingleClass()
<<void>> writeRefferenceClass()

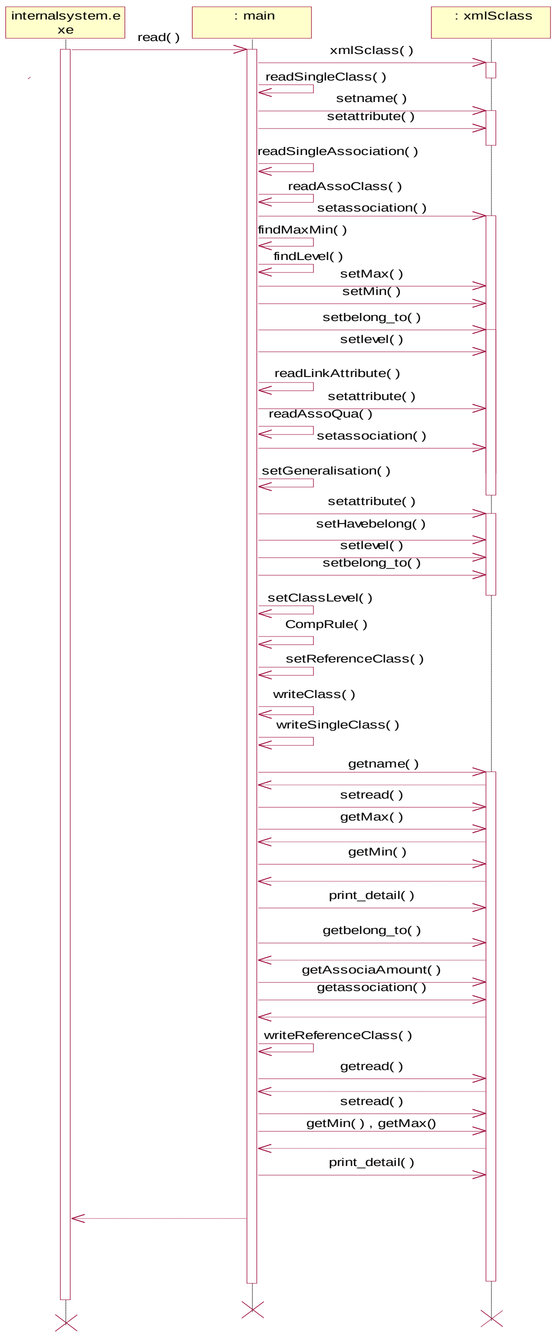
Class:	
Attribute&Operation	**Description**
<Attribute>	
dtdclass	The deque vector to contain the list of the class
<Operation>	

CompRule()	rule of compare the class
read()	This is main reading function to read the file, open the file and start read the file. Call the other read function to finish the some detail reading.
readSingleClass()	Read the each single class detail. When the new class appeared this function will be called.
readAssoClass()	Read class detail which include in the association relationship. When the classes appear in the association relationship, this function will be called.
readSingleAssociation()	Read a single association relation. When the new association appeared this function will be called.
readLinkAttribute()	Read the linker classes attribute detail inside of the association. When the new linker attribute appear, this function will be call to get the attribute detail.
readAssoQua()	Read the linker's qualification. When the qualification attribute appear, this function will be call.
findLevel()	Find the class level depend on the class appear times.
setGeneralisation()	Set the generalization of the class's detail. Copy the each attribute in the "father" class into the "son" class.
setClassLevel()	The class is 0..1 or 0..*, set the class be level 4
setReferenceClass()	Set the class is the reference class in the dtd file
getReferenceClass()	Find the reference class name and return the list of the class name. So that these names will occur in the root element declaration.
getISA()	Get the generalization class name.
writeClass()	Write the head detail into the DTD format document, and call the write single class to write the each single class.
writeSingleClass()	Write the class and it sub-class into DTD file
writeReferenceClass()	write the reference class into DTD file

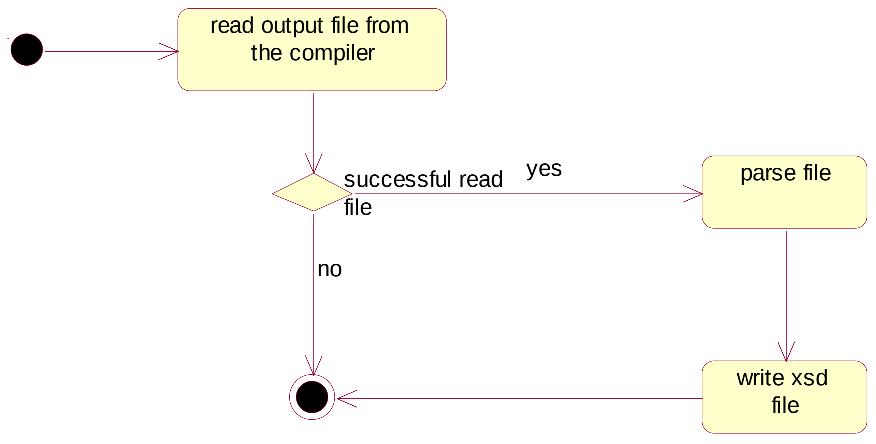
XML Schema Generate

Sequence Diagram

XML Schema

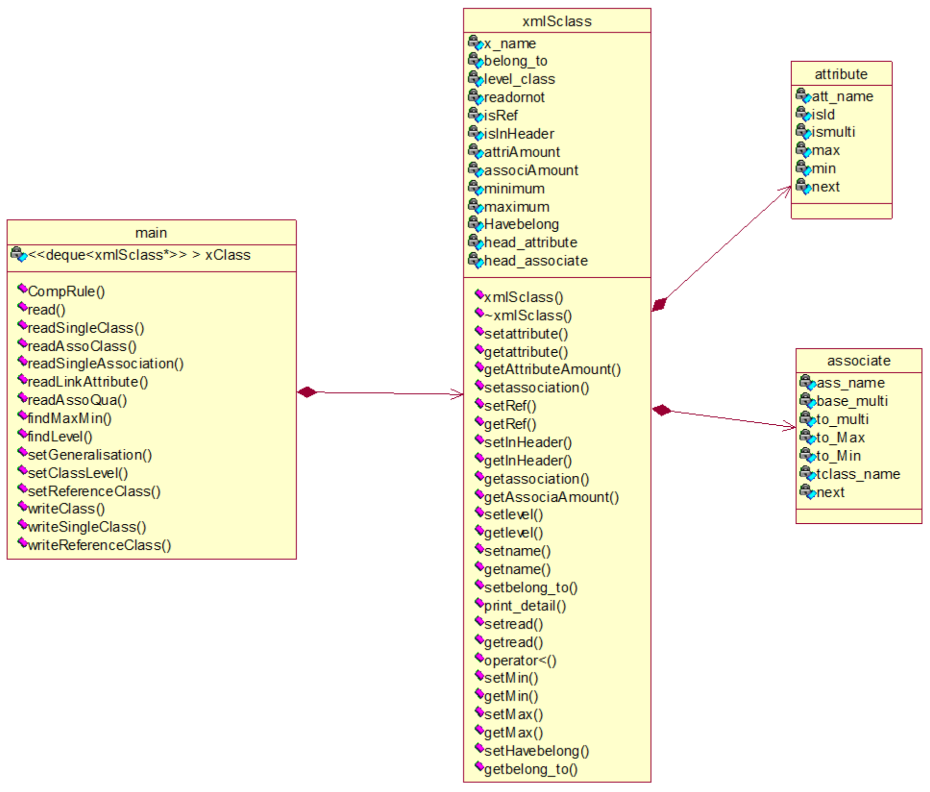


Statement Diagram



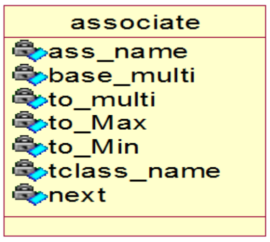
Class Diagram

XML Schema



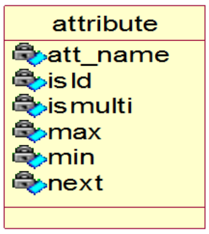
Class Description

Class: associate



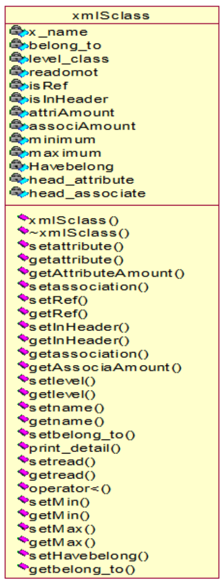
Class:	
Attribute&Operation	**Description**
<Attribute>	
ass_name	Record the name of the association
base_multi	Record the quantity of the class that appear in this association
to_multi	Record the quantity of the relate class that appear in this association
toMax	Record the relate class maximum appear in this association
toMin	Record the relate class minimum appear in this association
tclass_name	Record the name of the relate class
next	Link to the next association

Class: attribute



Class:	
Attribute&Operation	**Description**
<Attribute>	
att_name	Record the name of the attribute
isId	Record the attribute is ID or not
Ismulti	Record the attribute is multiple or not
max	Record the maximum appear
mix	Record the minimum appear
next	Link to the next attribute

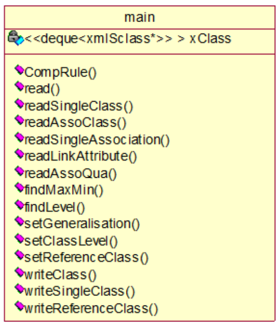
Class: xmlS



Class:	
Attribute&Operation	**Description**
<Attribute>	
x_name	Record the name of the class
belong_to	If the class has generation relationship with other class, record the name of "father" class.
level_class	Record the class level.
readornot	Record whether the class has written into the file already
isRef	Record whether the class is the reference class in the DTD file
isInHeader	Record the class whether has appeared in the other class
attriAmount	Record the number of attribute of the class
associAmount	Record the number of association of the class
minimum	Record the minimum appear
max	Record the maximum appear
HaveBelong	Record whether this class is a "father" class, have any other class belong to this class.
head_attribute	Link list of the attribute
prev_attribute	Record the current attribute
head_associate	Link list of the association
prev_associate	Record the current association
<Operation>	
xmlSclass()	Initializing the class
~xmlSclass()	Delete the class
setattribute()	Set or add the attribute into the class
getattribute()	Each time return the detail of one attribute in the class
getAttributeAmount()	Return the amount of the attributes

setassociation()	Set or add the association into the class
getassociation()	Each time return the detail of one association in the class
getAssociaAmount()	Return the amount of the associations
setRef()	Set the class is the reference class
getRef()	Return the class is the reference class or not
setInHeader()	Set the "isInHeader" attribute being true
getInHeader()	Return the "isInHeader" attribute state
setlevel()	Set the class level for sort inside of the deque, Class occurs 0 time in level 4, occur 1 time in level 1. Class occurs 1 time and the relate class occur more than 2 time set class in level 2. Class occur more than 2 times in level 3
getlevel()	return the level of the class
setname()	set the name of the class
getname()	return the name of the class
setbelong_to()	set the name of the class of the ISA relationship
getbelong_to()	return the name of the "ISA " class
print_detail()	Print the declaration of each class attribute into the file. But will not print the attribute which had appear before in the file.
setread()	set the class has been write into the DTD file
getread()	return the class has been write into the file or not
setMin()	Set the class minimum occurs time
getMin()	Return the class minimum occurs time
setMax()	Set the class maximum occurs time
getMax	Return the class maximum occurs time
setHavebelong	Set the class have other class depend to it

Class: Main



Class:	
Attribute&Operation	**Description**
<Attribute>	
dtdclass	The deque vector to contain the list of the class
<Operation>	
CompRule()	rule of compare the class

read()	This is main reading function to read the file, open the file and start read the file. Call the other read function to finish the some detail reading.
readSingleClass()	Read the each single class detail. When the new class appeared this function will be called.
readAssoClass()	Read class detail which include in the association relationship. When the classes appear in the association relationship, this function will be called.
readSingleAssociation()	Read a single association relation. When the new association appeared this function will be called.
readLinkAttribute()	Read the linker classes attribute detail inside of the association. When the new linker attribute appear, this function will be call to get the attribute detail.
readAssoQua()	Read the linker's qualification. When the qualification attribute appear, this function will be call.
findMaxMin()	
findLevel()	Find the class level depend on the class appear times.
setGeneralisation()	Set the generalization of the class’s detail. Copy the each attribute in the "father" class into the “son” class.
setClassLevel()	The class is 0..1 or 0..*, set the class be level 4
setReferenceClass()	Set the class is the reference class in the dtd file
getReferenceClass()	Find the reference class name and return the list of the class name. So that these names will occur in the root element declaration.
writeClass()	Write the head detail into the DTD format document, and call the write single class to write the each single class.
writeSingleClass()	Write the class and it sub-class into DTD file
writeReferenceClass()	write the reference class into DTD file

Team Composition

Group Member	Role	Responsible For
Yaowei Wang	Group Leader	Project design, documentation, graph generate
Xingfang Huang	Manager	GUI, XML scheme, DTD scheme
Jie Pei	Project Architect	Internal system, data dispose, SQL
Geliba	Core developer	Language compile, syntax check, home page
Zijun Liao	Core developer	Language compile, syntax check
Jie Zhang	Tester, Analyst	Graph generate, testing